VIMOS SOFTWARE FAMILY

Tool Description

20 February 2006

Thank you for your interest in our Vision Inspection and Measurement Optical System (VIMOS). In this manual you will find out detailed description of the VIMOS tools.

Before going on reading the manual, we kindly ask you to read the following

DISCLAIMER

THIS DOCUMENTATION IS PROVIDED FOR REFERENCE PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS DOCUMENTATION, THIS DOCUMENTATION IS PROVIDED “AS IS” WITHOUT ANY WARRANTY WHATSOEVER AND TO THE MAXIMUM EXTENT PERMITTED, ATTO-SYSTEMS LTD. DISCLAIMS ALL IMPLIED WARRANTIES, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE SAME. ATTO-SYSTEMS LTD. SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, DIRECT, INDIRECT, CONSEQUENTIAL OR INCIDENTAL DAMAGES, ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS DOCUMENTATION OR ANY OTHER DOCUMENTATION. NOTWITHSTANDING ANYTHING TO THE CONTRARY, NOTHING CONTAINED IN THIS DOCUMENTATION OR ANY OTHER DOCUMENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM ATTO-SYSTEMS LTD. (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF THIS SOFTWARE.

THE DESCRIBED SOFTWARE IS PROVIDED 'AS IS', WITHOUT ANY WARRANTY EXPRESSED OR IMPLIED. NO GUARANTY IS GIVEN THAT THE SOFTWARE IS SUITABLE FOR ANY GIVEN PURPOSE.

COPYRIGHT

TRADEMARKS

CONVENTIONS USED IN THIS MANUAL

***INFORMATION**. This sign marks section in the manual, which is for information only. You can decide to read or skip this section.*

***ATTENTION**. This sign marks section of the manual, which is particularly important for the general understanding of VIMOS. Please, make sure to read this section before proceeding with reading the manual.*

***TIPS & TRICKS**. This sign marks a Tips & Tricks section. Here you can find some practical advises on using the system or get a more detailed explanation of some features. Reading this section may help you in solving a particular problem or give you some ideas but is not vital for understanding VIMOS.*

***PREMISE**. This sign marks a section, which requires you to do something before proceeding with reading the manual. Usually this is a demo program, you have to run or something similar.*

File	*Menu item*
File > Open	*Sub-menu item or dialog control*
"1.1 About"	*Section reference. If the section is within the current manual no manual name is specified. When the section is within external manual the name of the respective manual is also included.*
`Ctrl+E`	*Hot-key combination. The first part of the combination specifies which system key to use. Possible values are: Ctrl, Alt, Shift. The second part specifies the normal key to be used in the combination. The plus sign means that you should press these keys simultaneously.*

CONTENTS

1. Introduction

This manual describes user-program tools in details. The description of each tool contains the following items:

- **Group** : A group the tool belongs to. Tools are organized in several groups depending on their functions. Sometimes a tool would belong to more then one group and then a compromise is used. The 'Send point-list' tool for instance is a member of the 'IO tools' group but could alternatively belong to the 'Point-list tools' group.
- **Short description** : Short tool description in several words
- **VIMOS kernel presentation** : Tool's graphical representation in VIMOS kernel
- **Editor icons** : Tool's representation in Editor
- **Description** : Detailed description of tool operation
- **Algorithm** : Tool algorithm (optional item, present for some tools)
- **Arguments** : Description of the tools arguments (inputs) in its configuration dialog
- **Results** : Description of the results of the tool
- **Similar tools** : List of tools with similar operation
- **Usually combined with** : List of tools, which are usually used together with the described tool
- **Tips & Tricks** : Additional information and useful advises.
- **Examples** : List of examples that use the tool and show its operation

Tool errors and colors

Each tool result has associated error code, which is set to non-zero value when the tool is not able to produce a valid result – for example when an edge-detection tool can't find an edge. Other errors could be caused by insufficient system resources - for example when a tool is not able to allocate enough memory and others.

Another general rule is: **If an argument of a given tool is linked to a tool result with error, then the current tool also produces error result(s).**

All tools with graphical representations in the overlay (with the exception of the GUI tools) have 3 standard error-context color arguments:

- **Draw_clr** : drawing color on success (no tool error). The default color 0 is green.
- **Err_clr** : drawing color on error (the tool has error results). The default color 0 is red.
- **Draw_mode** : tool drawing mode:
  - Always draw the tool
  - Never draw the tool
  - Draw the tool on error only
  - Draw the tool on success only

If a tool has no results, the colors are set according to the OK/error state of the arguments.

***ATTENTION**. Tool colors are available on PC and TI camera only.*

2. Image-processing tools

The tools in this group read or even modify pixels of current image, stored in the ‘Frame buffer’ (the actual image of VIMOS kernel. Some tools may access the ‘Freeze buffer’ - a secondary image mainly used as a scratch buffer but useful to store masks, templates and other image areas for quick use. The frame buffer is filled by the “Take image” tool or by automatic image acquisition, done at the beginning of each user-program pass (if not disabled in the video mode configuration). Set run-mode “Live & shoot” or “Shoot & show” to enable automatic image taking.

Editor toolbar of this group of tools :

2.1. Edge Detection

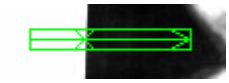
Group

Image processing tools

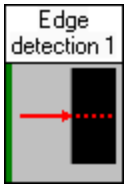
Short description

The edge-detection tool finds an edge between light and dark image areas.

VIMOS kernel presentation



Editor icons



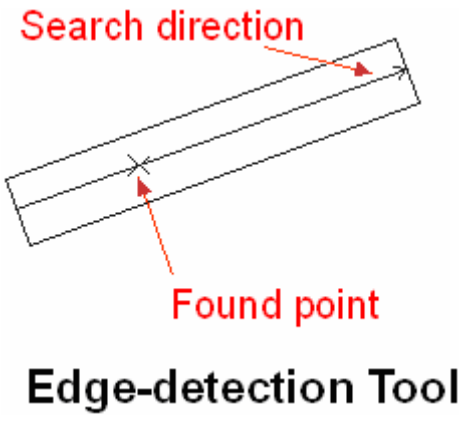
Description

General function :

The tool detects an edge between light and dark pixel areas in the current image, stored in the frame buffer. Two types of edges can be detected:

- light to dark edge
- dark to light edge

The tool uses an rectangular workspace that can be rotated. The edge is searched on the middle of the rectangle (axis).



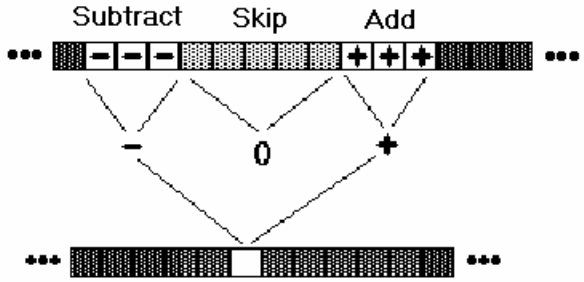
Results :

The tool returns the point of the edge or an error, if no edge was found.

Algorithm

Along the central axis of the rectangular workspace a single line is produced as an average of the brightness of all the pixels that are at the same row (left and right of the axis). Pixels outside the screen are excluded from calculations. By selecting a smaller or wider rectangle the user can influence the amount of the averaging. It is useful to ignore small ripple and noise at the edge but to be useful the edge detection has to be positioned orthogonal to the edge. The position of the edge on that central axis is then detected by finding the maximum of the first derivative of the brightness curve on this line (the place of maximum rising or falling of the brightness on the edge).

Digital differentiation is used. The algorithm moves along the axis and for every place generates the local rising or falling by subtracting the values of one group of pixels from the values of another group of pixels. You can define the size and location of the groups relative to the actual position of the calculation. In general the group of pixels that are subtracted is located behind the actual position while the group of pixels from which they are subtracted (in other words that are added) are located in front of the actual position. Since both groups are of equal size a positive value is generated if the area behind is darker then the area in front (dark to bright edge). The highest value is generated where the difference between both brightness values is at the maximum (black vs. white). The simplest case would be to have a group size of 1 (Calc. points) and a distance between groups of 1 (Diff. distance). Then the local result would be just the difference between two adjacent pixels (brightness values). If a higher value is used for 'Diff. distance' the subtraction is made with the brightness values of pixels that are at some distance and between both groups some pixels are ignored by the calculation of the local value of the rising. If 'Calc. points' is set to a higher value more then one pixel is used for adding and subtracting. This is helpful if you wish to middle out some local noise.



Arguments

Argument	Description
Point	Center point of rectangular workspace.
Angle	Angle of rectangular workspace
Width	Width of rectangular workspace
Height	Height of rectangular workspace
Dash length	Rectangle line style: 0 = solid line, non-zero = dashed line
Edge direction	Type of detected edge: Light -> Dark or Dark -> Light. The edge is searched in the direction of the arrow (left to right on initialization).
Diff. distance	Distance of differentiation: difference between indexes of pixel values, which are subtracted at differentiation. 'Diff. distance' = Skip + 1 = 6 on the figure above. The default difference value is 1, which means that adjacent pixels are subtracted: N1-N0; N2-N1;

	N3-N2, ... A value of 2 specifies: N2-N0; N3-N1; N4-N2, ...
Low limit	Minimum resultant value that gives valid edge or limit that must be exceeded. **Note:** Currently there is no compensation for 'Calc. points'. So if you change 'Calc. points' you should adjust this limit (sorry).
Keep limit	Specifies how many elements have to fulfill the previous criteria. The minimum value is 1. **Note:** If Keep limit is higher then 1 you will not be able to find perfect edges because they have their whole rising within only one pixel. So for good edges please always set Keep limit to 1.
Low pass filter	Low-pass filter parameter. A value of 1 disables the filter. The filtering is done by simple addition for greater values. For example a value of 3 determines the following filter: M0 = N0+N1+N2 M1 = N1+N2+N3 You may use higher values of 'Calc. points' for an equal effect.
Calc. points	Number of pixels in each one of the "subtract" and "add" areas. This is the number of pixels, which are included in the calculation with opposite signs. 'Calc. points' = 3 on the figure above.
Subpixels	Specifies pixel precision of calculations. A value of 1 specifies that calculations are done with precision of 1 pixel. Greater values (2 to 20) specify calculations with sub-pixel precision of ½, 1/3, …, 1/20 pixel.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Edge point	Detected edge point. It is located on the edge, crossed by the middle rectangle arrow.

Similar tools

Edge detection 2	uses external threshold value and writes results in the point-list buffer. This tool works on more then one line so that a lot of parallel edge points can be produced. It's possible to find the first, the last, the strongest or every edge.
Fast edge detection	works on just one line without average calculation (0,45,90, ... deg.).
Point-list edge	works on one parameter of the point-list elements within a given area of the point-list. More then one edge can be found and their places are again stored in another area within the point-list. A full sequence to use this feature would first generate pixel locations (coordinates of points – x,y) for each element of the point-list (i.e. tools 'Point-list circle' or 'Contour' or 'Load point-list'), would then read the brightness of the pixels at these locations and would then start 'Point-list edge' along on these values to find edges.

Usually combined with

Angle	
Distance	After finding the location of an edge you can start to measure.
Best line	If you have some points along an edge you can approximate it with a line.

Tips & Tricks

In many cases the object is moving a bit within the image. You would then want to compensate for this movement. In VIMOS ‘steering’ is used for that purpose. Tools are originally positioned relative to a given point and then moved, when this point moves. If you put horizontal edge detection from the left border to the left border of the object you get a point which can be used to steer all the other tools horizontally. For instance now another edge detection (vertical) can find the upper edge of the object at a given place – lets say 5 mm to the right of the left border. If the object moves to the right, then the first edge detection returns a result point more to the right and since the vertical edge detection is steered in X-direction by this point it moves the same amount to the right. Of course the result of that second edge detection can now be used for vertical steering of other tools.

Examples

Not indexed yet – sorry.

2.2. Edge Detection 2

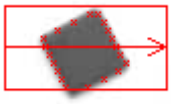
Group

Image-processing tools

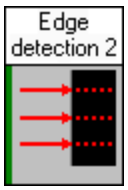
Short description

The tool uses to detect edges using an external brightness threshold. The tool's results are written in point list buffer.

VIMOS kernel presentation



Editor icons



Description

General function :

The tool is used to detect edges using an external brightness threshold. This threshold can be linked to results from other tools or set as a constant in the dialogue of the tool. The tool can find more then one edge. It searches for edges on a given number of parallel search paths and on every path it can return the first, the last, the strongest or all edges. The user can configure if he wants to find dark to light, light to dark or both kinds of edges.

The tool has a rectangular workspace that can be rotated.

Results :

The tool generates edge points, which are written in the point list buffer.

Additionally the first edge point found on the axis of the rectangle is returned as a normal result of the tool, so that it can be linked directly to inputs of other tools. If no such point was found the tool will return error for that result point, but will fill the point list buffer with edge points found on the other search paths.

The second point returned by the tool is a special point near the first one. It is always darker then the threshold. This point is used by the "Contour tool" which requires to be feed with a point that's brightness is below the threshold but is located next to a point with brightness above the threshold.

Algorithm

The tool performs simple location of the place where the interpolated brightness makes a transition through the threshold value. The edge point is given with sub-pixel accuracy, but the interpolation takes only two adjacent pixels into account.

Arguments

Argument	Description
Point	Center point of rectangular workspace
Angle	Angle of rectangular workspace
Width	Width of rectangular workspace
Height	Height of rectangular workspace
Dash length	Rectangle line style: 0 = solid line, non-zero = dashed line
Edge direction	Type of detected edge: Light -> Dark, Dark -> Light or Light -> Dark AND Dark -> Light. The edge is searched in the direction of the arrow (left to right at initialization).
Threshold	This brightness threshold can be linked (i.e. to a 'Percent threshold' tool) or set as a constant in the tools dialog.
Pt-list start	Start position for tool's result in point buffer
Num. of lines	Number of lines where the tool will search edges (set internally by the system to odd value, 1<= value <= "Height")
Get Edge	First, Last, Strongest, All edges
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Edge point 1	First edge point on the middle axis that fulfills the conditions given. So if only the last point was to be returned for every search path this result would be the last edge point found on the central line of the rectangle. This point is given with sub-pixel accuracy.
Number of points	Total number of points found – written to the point list buffer
Edge point 2	First edge point on the middle axis, but pointing to the real pixel that is the last/first directly below the threshold. Good starting point for the "Contour tool".
Brightness at edge point 2	The brightness value of P2 which may be used as an input to the threshold of the contour tool depending on the given task. In other cases it would be better to use the same constant threshold for the contour that was used for the 'Edge detection 2' tool.

Similar tools

Edge detection uses the first derivative to determine the place of the edge and not just a simple brightness threshold. It accumulates the brightness values of the pixels

	left and right of the search path and is therefore much more stable against local noise or rippling at the edge. The resulting location of the edge point will normally be of higher precision and stability. However this tool is slower and returns only one single point. To get some points along an edge you would have to place more then one such tool.
Fast edge detection	works like the 'Edge detection' tool, but just on one line without average calculation (0,45,90, ... deg.). Well, it's faster ☺.
Point-list edge	works on one parameter of the point-list elements within a given area of the point-list. More then one edge can be found and their places are again stored in another area within the point-list. A full sequence to use this feature would first generate pixel locations (coordinates of points – x,y) for each element of the point-list (i.e. tools 'Point-list circle' or 'Contour' or 'Load point-list'), would then read the brightness of the pixels at these locations and would then start 'Point-list edge' along on these values to find edges.

Usually combined with

Contour	This tool returns a special point on the edge of an object that is always located in a way that is suitable for the 'Contour' tool. It generates a sequence of points in the point-list buffer that surround the object.
Point-list best line	
Point-list best circle	If you have some points along an edge you can approximate it with a line or circle.

Tips & Tricks

Since this tool does not have an integrated filtering capability it might be advisable in noisy images to filter the area before the edge detection. For that purpose please use the tools '3x3 Operator' or 'Median Filter.

Examples

Not indexed yet – sorry.

2.3. Fast Edge Detection

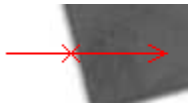
Group

Image-processing tools

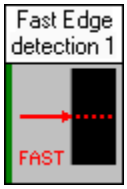
Short description

The tool is a faster version of the normal ‘Edge detection’ tool. It works only in certain directions and does not accumulate brightness values left and right of the path. So it’s faster, but not so strong.

VIMOS kernel presentation



Editor icons



Description

General function :

The tool finds edge points on one single straight line. Three different algorithms are provided.

Results :

The tool returns an edge point, which is located on the axis.

Algorithm

The user can select one of three methods for the edge detection. The first two methods are based on the “first derivative” and the formula

```
dG(y) = G(x) - G(x-N)
```

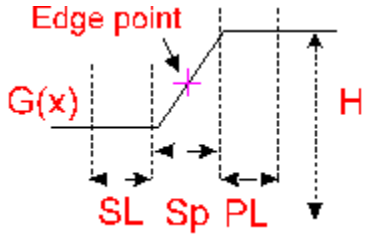
where:

- `x` is a position on the line of pixels along the search path (tools axis)
- `G(x)` is the gray value (brightness) at location `x`
- `y` describes the location exactly between ‘`x`’ and ‘`x-N`’ on the search path
- `dG(y)` is the first derivative of the brightness curve at position `y`
- `N` is equal to 1 for the first method and equal to two for the second

In other words calculating the difference between the brightness values of two close points generates the resulting value. In the first case two adjacent pixels are used, in the second case there is just one pixel between them. The edge point is located where the difference reaches its maximum absolute

value (if it is above the threshold). Please read the description of the normal 'Edge detection' tool for a more detailed explanation of the approach.

The third method uses a totally different way to determine the location of an edge :



- G(x) is the gray value (brightness) at location x
- SL is a number of pixels with no big differences in their brightness
- Sp is a number of pixels where the brightness changes more then a given threshold
- PL is a number of pixels with no big differences in their brightness

The algorithm detects an edge if it is high enough, happens within a given amount of pixels (SP) and is not generated by local noise. The latter is ascertained by the requirement of relatively stable brightness levels before and after the transition.

Arguments

Argument	Description
Point	Center point of the linear workspace
Angle	Angle of the linear workspace
Length	Length of the linear workspace
Dash length	Arrow line style: 0 = solid line, non-zero = dashed line
Edge direction	Type of detected edge: Light -> Dark or Dark -> Light. The edge is searched in the direction of the arrow (left to right at initialization).
Edge height thresh.	Depending on the selected method the local value of the first derivative or of the total brightness difference of the edge is required to exceed this value to produce an edge point.
Edge width thresh.	The number of consecutive pixels required to fulfill the requirement given by the previous parameter. An edge point is produced only if the first requirement is fulfilled for at least as many pixels as are given here. **Note:** When working with excellent edges and using the first or second method their quick rising from dark to light will generate just one high but tight peak on the first derivative. If you demand more than one pixel width of the edge then you will exclude such ideal edges that make the transition in just one pixels width. So better set this parameter to 1 in such a case.
Type of method	Selector for one of the three methods
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Edge point 1	The detected edge point

Similar tools

Edge detection — uses the first derivative to determine the place of the edge and not just a simple brightness threshold. It accumulates the brightness values of the pixels left and right of the search path and is therefore much more stable against local noise or rippling at the edge. The resulting location of the edge point will normally be of higher precision and stability. However this tool is slower and returns only one single point. To get some points along an edge you would have to place more then one such tool.

Edge detection 2 — uses external threshold value and writes results in the point-list buffer. This tool works on more then one line so that a lot of parallel edge points can be produced. It's possible to find the first, the last, the strongest or every edge.

Point-list edge — works on one parameter of the point-list elements within a given area of the point-list. More then one edge can be found and their places are again stored in another area within the point-list. A full sequence to use this feature would first generate pixel locations (coordinates of points – x,y) for each element of the point-list (i.e. tools 'Point-list circle' or 'Contour' or 'Load point-list'), would then read the brightness of the pixels at these locations and would then start 'Point-list edge' along on these values to find edges.

Usually combined with

Angle

Distance — After finding the location of an edge you can start to measure.

Best line — If you have some points along an edge you can approximate it with a line.

Tips & Tricks

In many cases the object is moving a bit within the image. You would then want to compensate for this movement. In VIMOS 'steering' is used for that purpose. Tools are originally positioned relative to a given point and then moved, when this point moves. If you put horizontal edge detection from the left border to the left border of the object you get a point which can be used to steer all the other tools horizontally. For instance now another edge detection (vertical) can find the upper edge of the object at a given place – lets say 5 mm to the right of the left border. If the object moves to the right, then the first edge detection returns a result point more to the right and since the vertical edge detection is steered in X-direction by this point it moves the same amount to the right. Of course the result of that second edge detection can now be used for vertical steering of other tools.

Since this tool does not have an integrated filtering capability it might be advisable in noisy images to filter the area before the edge detection. For that purpose please use the tools '3x3 Operator' or 'Median Filter'.

Examples

Not indexed yet – sorry.

2.4. 3x3 Operator

Group

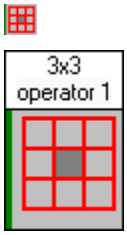
Image-processing tools

Short description

The operator applies a filter on a region of the image, defined by a rotated rectangle.

VIMOS kernel presentation

Editor icons



Description

General function :

The operator applies a filter on a region of the image, defined by a rotated rectangle. The filter is specified by a 3x3 matrix of **coefficients**:

```
| C11, C12, C13 |
| C21, C22, C23 |
| C31, C32, C33 |
```

and the **Shift value**. The new brightness of a given pixel is calculated depending on these values (see Algorithm) and then set to the pixel. The arguments **Point**, **Angle**, **Width**, **Height** and **Dash length** define the rectangular workspace. This tool has no results.

Algorithm

The tool calculates a new brightness for every pixel of the workspace. To do so the matrix of coefficients is positioned (virtually) ‘over’ the pixels of the image (frame buffer) in such a way that C22 is located ‘over’ the actual pixel while C11 is one row above and one column to the left (assuming an non-rotated workspace). Now the brightness of a pixel is multiplied with the coefficient ‘above’ it. The nine results are then added and the result of that accumulation is shifted left or right as much binary positions as given by the parameter. This could be expensed by the formula :

```
Vx,y = ( C11*P(x-1,y-1) + C12*P(x,y-1) + C13*P(x+1,y-1) +
         C21*P(x-1,y)   + C22*P(x,y)   + C23*P(x+1,y)   +
         C31*P(x-1,y+1) + C32*P(x,y+1) + C33*P(x+1,y+1) ) << Shift;
```

The shift operation is used as a fast substitute for multiply or division.

The idea of this standard way to filter images is, to calculate the new value of a pixel depending on the values of the pixel itself and the values of the pixels surrounding the pixel. Now depending on the aim of the filtering different weight can be given to the pixels. Since the brightness of nine pixels goes into the calculation of one pixel in most cases the result of the accumulation will be much bigger then just the 0..255 range that one pixel can take. So the result has to be normalized to that range. That requires an additional division or multiplication and is done the fast way with the shift operation. A shift to the right is equal to a division by 2, so if you do it twice it's equal to a division by four and so on. In the other direction (left shift) a multiplication by 2 is executed. So the following operation is done :

Shift parameter	Operation
...	
-4	Division by 16
-3	Division by 8
-2	Division by 4
-1	Division by 2
0	no change (0)
+1	Multiplication by 2
+2	Multiplication by 4
+3	Multiplication by 8
+4	Multiplication by 16
...	

To use this the best way it is necessary to make the total sum of coefficients equal to one of these numbers (0,2,4,8,16,...) and then use the shift parameter to get back to one. So if the sum of all the coefficients is i.e. 8 one would use a shift of –3 because 8>>3 is equal to 8/8 is equal to 1.

The algorithm uses short arithmetic calculations for better speed. So if you see wrong results for higher values of the coefficients there is probably an overflow and you should try to use lower values.

How to use this in practice ? There are many standard 3x3 filters – for example :

```
| 1   1   1 |
| 1   0   1 |   shift –3 is a simple way to reduce noise (but also sharpness)
| 1   1   1 |


| 1   2   1 |
| 2 -12   2 |   shift 0 is an edge-detector (returns bright edge-lines on darker background)
| 1   2   1 |
```

Arguments

Argument	Description
Point	Center point of the rectangular workspace
Angle	Angle of the rectangular workspace
Width	Rectangle width in pixels
Height	Rectangle height in pixels
Dash length	Rectangle line style: 0 = solid line, non-zero = dashed line
Shift value	Result shifting – positive values mean left shift (multiplication)
C11 – C33	Matrix coefficients
Draw_clr	Tool drawing color on success (0=default : green)

Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

This tool has no results.

Similar tools

Find blobs	can be used to binarize an image with two threshold values. To get this effect the 'Destructive' feature has to be activated. The result is a image, where all pixels with an original brightness between the two thresholds get assigned one brightness and all the remaining points get assigned another brightness.
Median filter	Applies 3x3, 5x5 or 7x7 median filter.
Filter point-list	Applies a median filter (low pass, softening) to a sequence of values located within a given parameter of a part of the point-list buffer. A standard way to use this would be first to fill the points of the elements of that part of the point-list with coordinates of pixels (i.e. by calling the 'Generate point-list circle' tool), then using the 'Point-list read pixels' tool to get the actual brightness into one of the parameters of these point-list elements and finally calling 'Filter point-list'. After that i.e. the 'Point-list Edge detection' tool can be used ...

Usually combined with

This tool is normally used for filtration before another tool from "Image processing" group as i.e. : Fast edge detection, Edge detection 2, Find blobs, Contour, Barcode, OCR and others.

Tips & Tricks

This tool is not one of the fastest because it requires a lot of memory access and calculations. If you can use it non-rotated the speed may be much better then otherwise. It is always recommended to use the smallest possible workspace for best speed. Eventually it may be possible to use the 'Pyramid' operation of the 'Image area' tool to reduce the size of the area that needs to be filtered. Since this operation has a noise reducing effect it may even improve the image enough to make the use of the 3x3 filter unnecessary.

If you need to preserve an area you need to filter you can use the 'Copy' operation of the 'Image area' tool and copy an area to another place or to the Freeze buffer. After you finished with the filtering and the following tasks you would be able to copy the original back or to process it at its new place.

Examples

Not indexed yet – sorry.

2.5. Light Balance

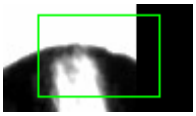
Group

Image-processing tools

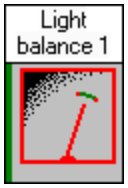
Short description

This tool monitors the brightness of a specified region of the image. If the illumination fails somehow the tool will notice that the average brightness of its workspace moved out of given thresholds and return a ‘bad’ result. This could also be used for other purposes like presence or absence or even an approximate position when an object masks a light source.

VIMOS kernel presentation



Editor icons



Description

General function :

This tool monitors the average brightness of a specified region of the image. The region is defined by a rectangle, which may be rotated.

Results :

There is just one result :

- -1 : the measured brightness is below the specified range
- 0 : the measured brightness is in the specified range
- +1 : the measured brightness is above the specified range

Algorithm

The tool accumulates the brightness values of all the required pixels within its rectangular workspace. A parameter (Step) tells which pixels should be accessed. If ‘Step’ is one every pixel is used. If it’s two just every second pixel in X and Y is used – making the process four times faster. After accumulating all the brightness values the average is generated by dividing with the number of pixels used. The result is compared to the upper and lower thresholds.

Arguments

Argument	Description
Point	Center point of the rectangular workspace
Angle	Angle of the rectangular workspace
Width	Rectangle width in pixels
Height	Rectangle height in pixels
Dash length	Rectangle line style: 0 = solid line, non-zero = dashed line
Step	Step in X and Y direction. A step, equal to 2, means to skip 1 row and 1 column every time.
Brightness	Expected brightness. It is updated to the current brightness by the **Get** button in the dialog. To get correct nominal value, video page 0 of the system must contain valid image, stored there by previous operation of the system in run-modes “Live & shoot” or ”Shoot & show”.
Up tolerance	Allowed positive tolerance of the brightness.
Down tolerance	Allowed negative tolerance of the brightness.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Brightness check	-1 = below lower threshold, 0=OK, 1 = above upper threshold

Similar tools

Percent threshold — returns a brightness value at a given point between the lowest and the highest brightness of any pixel in the workspace. Since it has different behavior then the averaging of the ‘Light balance’ tool it may be an alternative, but better use a filter if your images are noisy.

Rectangle/Circle test — these tools return the percentage of the number of pixels (area) with brightness above the threshold to those with brightness below the threshold. Again these tools do something different then averaging the brightness and may be an alternative to the ‘Light balance’ tool.

Usually combined with

Normally the tool is used ‘stand alone’ to make sure the illumination is not changing to much for all the other tools to work normally.

Tips & Tricks

Just place some of these tools in quite corners of your workspace and when all the illumination is set up finally teach these tools by hitting the ‘Get’ button in their dialogues. Then combine their outputs

with an IF and generate an error, if one of the tools returns ‘bad’. Maybe someone had his head between the lights and the workspace ... Since these tools need processing time, you’ll have to find out how much of the tools you can use and what ‘Step’ you have to use.

Of course you can detect the presence or even the approximate position of objects that mask the illumination. It is even possible to use the tool as a ‘fill level indicator’ or similar.

Examples

Not indexed yet – sorry.

2.6. Percent Threshold

Group

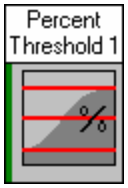
Image-processing tools

Short description

This tool returns a brightness value between the highest and the lowest brightness in its working area. The exact location between the maximum and minimum brightness is given as a parameter in percent. It is possible to request two other work modes where either zero is used for the lower value or 255 is used for the higher value.

VIMOS kernel presentation

Editor icons



Description

General function :

This tool determines the minimum and maximum pixel intensities (brightness) (I_{min}, I_{max}) within a given rectangle and returns and intermediate intensity (I) based on a given percentage (P).

There are three modes of operation of this tool:

- **`MIN-MAX`** - normal (use min and max. intensities from the image)
- **`000-MAX`** - assume $I_{min} = 0$
- **`MIN-255`** - assume $I_{max} = 255$

Results :

Calculated intensity (I) and average (A).

Algorithm

The tool uses the formula : $I = I_{min} + (I_{max} - I_{min}) * P / 100$, where

I = intensity

P = percentage

Arguments

Argument	Description
Point	Center point of the rectangular workspace
Width	Rectangle width in pixels
Height	Rectangle height in pixels
Angle	Rotation angle (currently not used)
X-step	Horizontal step to skip pixels (columns).
Y-step	Vertical step to skip pixels (rows).
Mode	MIN-MAX / 000-MAX / MIN-255 (see description and algorithm)
Percentage	Percentage (P) to be used (see description and algorithm)
Dash length	Rectangle line style: 0 = solid line, non-zero = dashed line
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Intensity	Calculated intensity (I) (brightness)
Average	Calculated average value (A)

Similar tools

Edge detection 2 to find a threshold for the ‘Contour’ tool. Please read the chapter of these tools to find out how to use them in combination.

Usually combined with

3x3 filter

Image projection Noise in the image is easily limiting the usefulness of the ‘Percent threshold’ tool. The tools mentioned here can smoothen the image a bit to have a better input for the ‘Percent threshold’ tool. The ‘Image projection’ tool is faster then the ‘3x3 filter’ tool, but it has its limitations.

Find blobs to generate thresholds for the binarization

Edge detection 2

Point-list edge to generate the edge threshold

Tips & Tricks

Use this tool to create programs that are relatively stable against changes of the actual image brightness.

Examples

Not indexed yet – sorry.

2.7. Rectangle Test

Group

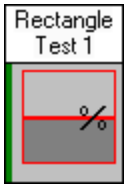
Image-processing tools

Short description

This tool calculates the proportion between pixels darker or brighter then a given threshold within a given rectangle.

VIMOS kernel presentation

Editor icons



Description

General function :

This tool calculates the proportion of dark and bright pixels within a given rectangle. All pixels with brightness lower or equal to the given threshold are counted as dark. Pixels with brightness above the threshold are considered bright.

Results :

The tool returns the percentage of dark pixels and percentage of bright pixels.

Algorithm

For every pixel in the workspace check if the brightness is above the threshold or not and then increment the respective counter. Then divide the final values of both counters by the sum of their values and return both results.

Arguments

Argument	Description
Point	Center point of the rectangular workspace
Width	Rectangle width in pixels
Height	Rectangle height in pixels

Angle	Rotation angle (currently ignored)
Threshold	Brightness level that divides between dark and bright
Dash length	Rectangle line style: 0 = solid line, non-zero = dashed line
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Dark part	Percentage of dark pixels (0-100)
Bright part	Percentage of bright pixels (0-100)

Similar tools

Circle test does the same in a circular workspace

Usually combined with

Normally the tool is used ‘stand alone’ to determine presence or absence of some objects within the area or to check the quality of their surface. The result of the tool is used to make decisions.

Tips & Tricks

This tool is quite stable against noise in some situations and may provide a fast method to solve positioning and absence/presence tasks.

Examples

Not indexed yet – sorry.

2.8. Circle Test

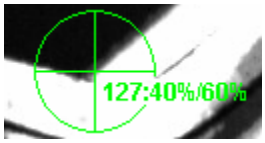
Group

Image-processing tools

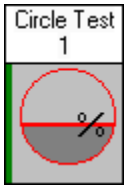
Short description

This tool calculates the proportion between pixels darker or brighter then a given threshold within a given circle.

VIMOS kernel presentation



Editor icons



Description

General function :

This tool calculates the proportion of dark and bright pixels within a given circle. All pixels with brightness lower or equal to the given threshold are counted as dark. Pixels with brightness above the threshold are considered bright.

Results :

The tool returns the percentage of dark pixels and percentage of bright pixels.

Algorithm

For every pixel in the workspace check if the brightness is above the threshold or not and then increment the respective counter. Then divide the final values of both counters by the sum of their values and return both results.

Arguments

Argument	Description
Center	Center point of the circular workspace
Radius	Radius of the circular workspace

Threshold	Brightness level that divides between dark and bright
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Dark part	Percentage of dark pixels (0-100)
Bright part	Percentage of bright pixels (0-100)

Similar tools

Rectangle test does the same in a rectangular workspace

Usually combined with

Normally the tool is used ‘stand alone’ to determine presence or absence of some objects within the area or to check the quality of their surface. The result of the tool is used to make decisions.

Tips & Tricks

This tool is quite stable against noise in some situations and may provide a fast methode to solve positioning and absence/presence tasks.

Examples

Not indexed yet – sorry.

2.9. Find Blobs

Group

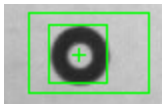
Image-processing tools

Short description

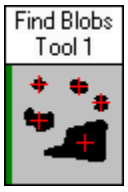
'Blob' is abbreviation for '**b**inary **l**arge **ob**ject' – still not very plain to understand. The idea is, that using some criteria you divide all the pixels of the work space of the tool into two groups – let's say dark and bright or simply '0' and '1'. After that you assume, that '0' means 'background' and '1' means 'foreground'. Now there will be groups where foreground pixels cluster together surrounded by background pixels. These are the objects. Such objects have features like size (number of pixels), position of the center of mass and others.

This tool uses an upper and a lower brightness threshold and all the pixels with brightness between both thresholds are foreground (belonging to objects) while the others are background ones. The tool is able to fill both areas with a given brightness. The tool returns the features of all the objects found as entries to the point-list buffer.

VIMOS kernel presentation



Editor icons



Description

General function :

This tool works in a rectangular workspace that at the moment should NOT be rotated because that functionality is not implemented yet. First the brightness of every pixel is compared to the thresholds given. Then the tool acts depending on the configuration :

Destructive / Nondestructive :	If 'Destructive' is selected the image is filled with two shades of gray – depending on what the user selected for the blobs and the background
Connected / Unconnected :	If 'Connected' is selected in a second step objects are build by finding groups of neighboring foreground pixels that are completely surrounded by background pixels. Then results are produced for every such object (blob). If 'Unconnected' is chosen just one set of results is produced that is corresponding to one big foreground object containing all the foreground pixels.

Area filtering : In the 'Connected' mode the tool can ignore too small and too large objects. They become part of the background. This feature may be used to ignore noise or to distinguish between different known parts of the image that are of different size.

Important - please read next limitations on ADSP-based cameras !

***ATTENTION**. Limitations on ADSP-based camera in **connected** mode:*

- *The maximum number of objects that can be processed is limited to **500**. The tool returns error 9005 in case of object overflow.*
- *The maximum width of the work area is about **540** pixels (no limitations for the height of the work area). The tool returns error 2 (memory allocation error) in case of too large width.*

If more objects are actually in the image, then no objects are returned. The problem is that we are talking about the number of objects BEFORE the area filtering. So if the image is noisy you may have zillions of small objects of just one or two pixels and the tool may fail. To see this you could switch to the 'Unconnected' and 'Destructive' mode of work and see all the foreground pixels highlighted because in that mode no object segmentation is attempted and the tool does not fail.

If you have such a situation it may be possible to smooth the image with a '3x3 filter' tool and take away the noise. Or it may be possible to divide the workspace and use more then one 'Find blobs' tool. If nothing helps you should use VIMOS on a TI-processor based camera. On the PC this effect is not happening, but better check the object count before ordering an ADSP-based camera.

Results :

For every object a set of results is produced : center of surrounding rectangle, width, height, area, center of mass and a point on the periphery of the object. This information is stored in the point-list buffer. One point-list item is filled for every object. Every point-list item has place for one point and eight parameters. To store all three points two pairs of parameters are used to store x and y coordinates. The 'Get blob' tool may be used to read one such item in the correct way. Another possibility would be to use 'Make point' tools to restore points out of pairs of coordinates. Please read about the point-list buffer for more information.

Point-list item usage :

Item field	Object data
Point	Bounding rectangle center point
Parameter 0	Bounding rectangle width
Parameter 1	Bounding rectangle height
Parameter 2	Bounding rectangle orientation angle
Parameter 3, Parameter 4	Center of mass (x,y)
Parameter 5	Object area (pixels)
Parameter 6, Parameter 7	Peripheral point (x,y) – rightmost point in the lowest row (from contour)

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Center point of the rectangular workspace
Angle	Angle of the rectangular workspace (DON'T USE)
Width	Rectangle width in pixels
Height	Rectangle height in pixels
Dash length	Rectangle line style: 0 = solid line, non-zero = dashed line
Max to find	Maximum objects to find in the region (to improve speed)
Connected	Method of searching objects – please see description
Destructive	Fill objects/background with two shades of gray
Min area	Minimum area of an object – objects with an area below this value are ignored. They become background in destructive mode.
Max area	Maximum area of an object – objects with an area above this value are ignored. They become background in destructive mode. **Note:** If this value is 0 there is no upper limit.
Down threshold	Lower brightness threshold – objects have to be brighter
Up threshold	Upper brightness threshold – objects have to be darker
Color of back	Color of background, used in destructive mode
Color of blob	Color of objects, used in destructive mode
Pt-list start	Start index in the point-list where object items will be stored
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Count	Number of objects found

Similar tools

There is nothing similar at the moment. Image segmentation by watersheds or region growing is in the making.

Usually combined with

3x3 filter — To reduce the number of random small objects generated by noise it may be necessary to filter the image with a low pass filter (smoothening).

Point-list sort

Point-list filter — The results of the tool are written to the point-list buffer. If 'Connected' mode was selected all the objects found get their own item in the point-list. It is then

	possible to sort (reorder) or filter them depending on one of their parameters or coordinates. So it's possible to find i.e. the biggest, the most left or the one with the lowest vertical extension.
Contour	Since the tool returns a starting point for contour following for every object in the 'Connected' mode it is a standard way to get the contours of objects.
Other tools	Since the blob tool returns the location of an object it is often used as an input for steering (relocating) other tools to that position, so that these tools can do their work at the right location relative to the object. This may be used for image processing or for example to place text near the object the text belongs to.

Tips & Tricks

This tool is usable for quite a lot of tasks like filtering, counting, measuring, masking, binarization ...

To binarize an image you would just use the tool in the 'Unconnected' and 'Destructive' mode. If you don't need two thresholds you can set the lower threshold to zero or the upper threshold to 256. If you would like to change the brightness of just a part of the pixels, but leave others intact you could make a copy of the workspace (to the Freeze buffer) using the 'Image area' tool and then call 'Find blobs'. After the binarization you could call 'Image area' again with the 'Maximum' or 'Minimum' function. That would restore the original grayscales of all the background or all the foreground pixels leaving the other 'half' of the pixels at the brightness you assigned to them with the 'Find blobs' tool.

It is worth noting that the center of mass of a blob is very accurate and stable against noise and small deformations of the blob. If you need to measure the distance between two objects this would be your first choice. You would have to use the 'Connected' mode and try to set the area filtering in a way that you get just the two objects you are seeking. If it is not possible that way, you can filter the results with other tools like 'Point-list filter' i.e. for position or other features.

Examples

Not indexed yet – sorry.

2.10. Optical Character Recognition (OCR)

Group

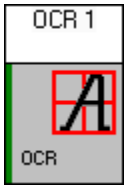
Image-processing tools

Short description

The tool is used to recognize characters or other symbols in a rotated rectangular workspace.

VIMOS kernel presentation

Editor icons



Description

General function :

The tool first calculates a binarization threshold and binarizes the work area (makes it black and white instead of gray). Alternatively the binarization can be done with other tools before this tool is called and the initial binarization may not be executed. After that the binary image is segmented in rectangular boxes that are expected to contain one letter (symbol) each. This is done by first detecting the vertical extension of the line(s) and then for every line the horizontal extension of the letters of that line. The next step is the processing of all the rectangular boxes. The content of every box is compared to all the templates (stored expected patterns of known symbols). The result of the process is one code for every box. It's the code that was assigned to the pattern that was the best match for the content of the box. The comparison (matching) is done by dividing the box into a certain number of squares and comparing the color (black or white) of each such square with the color of the same square in the template. This makes the algorithm independent of size, so that the same symbol can be detected in different sizes if the font is the same or at least close. A limitation of the actual property of the segmentation process is that it divides wherever there is a complete white vertical space between black things. So letters that overlap get in one box, while letters that contain a vertical division, get divided into two boxes.

This kind of OCR is of course not very strong. It is not intended for use in a noisy environment or on three-dimensional letters. However it proved its value for reading well-printed labels or other good printing. Since the symbols are teachable there is no direct limitation to a given set of symbols. It's possible to interpret non-standard signs or special characters.

It's necessary to note, that close symbols (and that depends on the font that is used) can not be separated. For example there is a good chance for errors with : [0,O], [1,I], [2,Z], [6,G] depending on the font. Normally there should be no problem to read numbers. A problem could be [5,6,8,9,0] if they are realized in an unlucky way. No our tests proved, that this is rather seldom. A lot of letters are also quite readable while it is much harder if numbers and letters can be found at the same place. If it's known, where a number is expected and where a letter post-processing can correct such errors.

The results of the tool are written to the string buffer as a sequence of codes. A '~' is inserted for all the unknown symbols.

Before using this tool you must create a data base file with the symbol information (templates) as described here :

- Make an image with a sequence of the following characters black on a white background :

  0 1 2 3 4 5 6 '7 8 9 A B C D E F G H I J

  K L M N O P Q R S T U V W X Y Z & % $ #

  (You can put all the symbols in one line or make a lot of lines. The best way would be to make a box out of the symbols so that they fill the screen of the camera without being too close one to the other in both directions. Please make sure you teach under the same angle as you are going to use for the OCR later.)

  This is the default series of the characters that is needed to get direct ASCII translation (see "APPENDIX A. ASCII table"). If you teach by using a different series of symbols the first one gets assigned the code 0x30 (30 hex), the second 0x31 and so on. It works with fewer symbols then shown above. The drawback is that the assignment will not be ASCII compatible if you teach with another then the standard sequence. In such a case the 'Show string' tool will not display what you would expect and some other tools that interpret the content of the string buffer will return 'strange' results to. In simple cases however such an approach may be used to save time.
- Place the printout of the symbols in front of the camera and adjust the image
- Add the OCR tool to the user program
- Use Move button, Width and Height spins to enclose the characters
- From 'Mode – selector' select 'Learn-mode'
- Hit the OK button of the dialog which executes the tool one time and does the teaching
- Directly go back into the Dialog (use 'Configure')
- Change the 'Mode selector' from 'Learn-mode' to 'Recognize'
- Exit from the dialog with OK again

After this procedure the OCR tool is ready to recognize characters. Please note, that if you execute the teaching directly on the camera the template database is written to the flash memory. Such a write always looses free space on that memory and it is possible to fill it completely. In such a case you will get an error message during the teaching and you will not be able to save your user program. At the moment there are no means integrated in VIMOS that would allow you to fix this problem. Just by using other tools directly on system level will you be able to reclaim free flash memory. Please read more about that situation in the 'Resources' manual. We are working to improve this situation.

At the moment it would be better to do the teaching on the PC (in the Simulator) where there is enough space for unlimited experimenting. Just after creating a good database you could transfer it as a file to the flash memory of the camera and use it there.

You can correct templates of symbol characters, or add new templates. Please use the following steps:

- Make an image of the new symbol
- Use Move button, Width and Height spins to enclose the symbol
- Select 'Correct char' or 'Add new char' at the 'Mode – selector'
- Select the code of the symbol that will be corrected with 'Char. Number selector'. A new symbol will always be added at the end of the current sequence. You can NOT 'correct' a symbol that had never been taught. An attempt to 'correct' an unused code will not work.

- Hit OK to execute the teaching once and directly return to the dialog with ‘Configure’
- Change the ‘Mode selector’ back to ‘Recognize’
- Exit the dialog with another OK and the tool is again ready to interpret symbols

In mode ‘Delete all’ the tool performs only one operation – it discards the existing template database and generates a new empty database. You may add characters to the database in mode ‘Add new char’.

You must satisfy the following requirements:

- min character’s size : 5x10 pixels
- maximum number of rows : 7
- maximum number of symbols (recognition) : 20
- min white border around the symbols : 2 pixels
- min white border between rows : 5 pixels
- min white border around all text : 5 pixels

Results :

The tool writes the symbol codes into the string buffer (see above) and returns the length of the recognized string (number of symbol codes written).

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Center	Center point of the rectangular workspace
Width	Rectangle width in pixels
Height	Rectangle height in pixels
Angle	Rectangle rotation angle
Line style	Rectangle line style: 0 = solid line, non-zero = dashed line
Mode	There are the following modes : • **Recognition** – the tool recognizes symbols and returns codes • **Learn** – tool teaches symbols and saves the database file. **The tool has to be started just one time in that mode. The mode should therefore better not be activated when the program is started in run-mode. This would sooner or later fill the whole flash memory, which is dangerous. Please read the description chapter above.** • **Correct char** – in this mode you can renew the template for the symbol with the code selected in the ‘Char number’ box. **This mode should NOT be selected when the user-program is started in run-mode because it will fill up the flash. Please read the description chapter above.** • **Add new char** – in this mode you can add the template of a new symbol to the first free code at the end of the queue. **This mode should NOT be selected when the user-program is started in run-mode because it will fill up**

	the flash. Please read the description chapter above. • **Delete all.** Deletes the template database file ***chini.vm*** (if present) and creates a new file with 0 symbols. You may add symbols in the file by the "**Add new char**" option.
Char number	This box is used only in 'Correct char' mode to select the code of the symbol that has to be replaced (corrected).
Debug mode	• Off – no debug information is displayed • On – the tool displays debug information in the upper left screen corner **Note :** In the Simulator (on the PC) this mode is always On.
Start position	Position in the string buffer where the first result code is written to
Simple binarization	Binarization mode: • Off – no binarization is done, tool expects black & white image • On – binarization is done, tool expects bimodal gray image
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Strlen	Number of codes written to the string buffer

Similar tools

We'll provide a more powerful OCR/OCV as soon as we can. The 'Barcode' tool reads barcodes ☺.

Usually combined with

Find blobs — can be used for external binarization with two thresholds that can be dynamically positioned by other tools (i.e. 'Percent threshold'). Another use for this tool in connection with the OCR tool could be the positioning of the OCR workspace over the symbols because if you run the 'Find blobs' tool in its 'Unconnected' mode it will return the center point of the non-rotated bounding box around all the objects with a brightness between the two thresholds. In some cases the center of gravity (center of mass) which is also returned may be of better use to position the OCR tool.

String tools — for the OCR results : Show string, Compare string, String to number

Tips & Tricks

This tool is not so easy to use and only some praxis will lead to good results. The symbols should not be to small and if possible a font should be used that has visible differences in the form (geometry) of all the symbols that may occur at a given location. Good angular alignment is also necessary. The tool should always be rotated in such a way, that its lower border is exactly parallel to the row of symbols.

Examples

Not indexed yet – sorry.

2.11. Bar Code Reader

Group

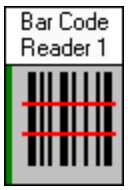
Image-processing tools

Short description

This tool reads a barcode that is located around the pointer of the tool. Actually the tool reads barcodes of the types ‘3 of 9’ and ‘2 of 5 interleaved’

VIMOS kernel presentation

Editor icons



Description

General function :

The tool starts seeking a valid barcode around a given point. The barcode can be rotated, but has to be as specified by the standard : dark on bright, with a considerable bright ‘quite zone’ at the left and the right side and some bright space above and below the barcode. This space has to be visibly wider then the maximum distance between two lines of the code. The code INCLUDING the full ‘quite zones‘ has to be within the image. Of course it has to be of the selected type.

After the dimensions of the code have been established the code is interpreted and the symbols encoded in it are written to the string buffer.

Results :

The tool writes the symbol codes into the string buffer and returns the length of the recognized string (number of symbol codes written).

Algorithm

After finding the area of the barcode three scan lines are put over the code. Along them the pattern is binarized with a dynamic threshold to get three sequences. If their interpretation gives a valid result it is returned. To be valid at least two of the three scans have to have the same correct sequence.

Arguments

Argument	Description
Point	Defines the position of the search algorithm starting point
Type of barcode	Defines the expected type of the barcode pattern
Start position	Position in the string buffer where the first result code is written to
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Count	Number of codes written to the string buffer

Similar tools

There is a small OCR tool. Please ask for other barcode types or for our ‘Data Matrix code’ tool.

Usually combined with

Find blobs — can be used for external binarization with two thresholds that can be dynamically positioned by other tools (i.e. ‘Percent threshold’). Another use for this tool in connection with the ‘Barcode’ tool could be the positioning of the pointer over the barcode because if you run the ‘Find blobs’ tool in its ‘Unconnected’ mode it will return the center point of the non-rotated bounding box around all the objects with a brightness between the two thresholds. In some cases the center of gravity (center of mass) which is also returned may be of better use to position the ‘Barcode’ tool.

String tools — for the result processing : Show string, Compare string, String to number

Tips & Tricks

The tool requires a certain degree of image resolution. Since the encoding of barcodes is hidden in the relative widths of the bars the image must transfer this information in as good a quality as possible. The better the printing, the illumination, the optical system and the physical resolution of the camera (mainly proportional to the image size) the more bars can be cleanly distinguished in the image and the longer a barcode can be interpreted. The human eye will have a better impression of the image then the camera has, because the noise of the life image helps it to fetch additional information out of the sequence of many different images where the tool ‘sees’ only one of them. Another advantage of the human eye is, that is sees the bars fully in two dimensions while for speed reasons a sub-pixel precision 2D image binarization with corresponding local thresholds and filtering is out of the question. So in reality there are some limitations to the maximum possible length of the barcode. Please don’t forget to include the ‘quite zones’ fully in the image. The tool has proven useful in practical cases, so the length limitation is not so bad.

It’s necessary to be noted that the tool mainly detects edges at the front and the rear of the bars to generate their width. While dynamic adjustments to the binarization threshold are done along the scan lines these adjustments have naturally to be kept much slower then the rising and falling at the edges

occurs. The consequence is, that hard shadows with edges that are steeper then a given limit can not be compensated and create invalid readings. It is highly recommended to assure that such shadows can not disturb the image.

There may be situations when the tool is almost always returning the correct result, but from time to time it fails to read a code. If system timing permits, we suggest putting a second ‘Barcode’ tool behind the first that is only executed when the first did not return the expected result. In such a case the starting point should be displaced by some pixels in both directions. If this is still not good enough a new image could be taken between both tools.

Examples

Not indexed yet – sorry.

2.12. Image Projection

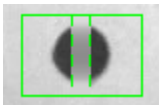
Group

Image-processing tools

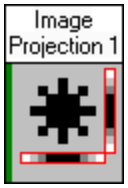
Short description

The tool reads all the brightness values of the pixels of one row or column of the workspace and finds their average value. A certain part of that row or column is filled with this average brightness. The process is repeated for all rows or all columns of the workspace of the tool.

VIMOS kernel presentation



Editor icons



Description

General function :

(Same as ‘Short description’) The tool reads all the brightness values of the pixels of one row or column of the workspace and finds their average value. A certain part of that row or column is filled with this average brightness. The process is repeated for all rows or all columns of the workspace of the tool.

Note : This tool works only in run-mode.

Results :

There are no other results then the changes in the image (frame buffer).

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Center point of the rectangular workspace

Width	Rectangle width in pixels
Height	Rectangle height in pixels
Angle	Rotation angle (currently ignored)
Direction	Horizontal / Vertical projection
Proj. width	Projection band width
Dash length	Rectangle line style: 0 = solid line, non-zero = dashed line
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

This tool has no results.

Similar tools

3x3 filter

Find blobs — These tools have in common, that they are used to change a part of the image depending on the former content. With the '3x3 filter' tool a lot of different effects can be created, but they are always very local since the tool 'sees' only 9 pixels. The 'Find blobs' tool can be used to binarize the image with one or two threshold values.

Usually combined with

Fast edge detection — The 'Fast edge detection' tool doesn't look at the pixels left or right of its search path for speed reasons. If the edges are noisy it would be possible to use the normal 'Edge detection' tool, but that is slow because it can be rotated freely and therefore samples the pixel values with sub-pixel routines. For horizontal or vertical edge detection the 'Fast edge detection' tool can be combined with the 'Image projection' tool, which is quite fast. The image projection tool does a noise limiting integration of all the values within a certain width so that a clean and stable edge is created.

Percent threshold — The 'Percent threshold' tool is of good use in noiseless images. It returns a value between the highest and the lowest brightness value in its workspace. But noise is very quickly making that tool useless, because it will always generate a white and a black pixel somewhere, so that always the value between 255 and 0 is returned – independent of the average brightness of the workspace. In such cases it may be useful to use a fast 'Image projection' tool that eliminates noise, but will otherwise leave the basic function of the 'Percent threshold' tool intact.

Tips & Tricks

There's nothing special to say about this tool.

Examples

Not indexed yet – sorry.

2.13. Contour

Group

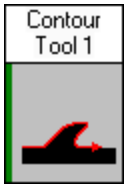
Image-processing tools

Short description

This tool starts at a given pixel that should be at the border between bright and dark areas (relative to a given threshold value) and tries to 'walk' along that border in clockwise or counterclockwise direction. All the pixels that the tool 'steps on' are considered to be the contour (of an object) and their location is written to the point-list buffer. To tool stops when the contour is closed or when it ends. The tool works with sub-pixel precision and it is possible to make steps shorter then one pixel by inserting a given number of sub-pixels between two adjacent pixels.

VIMOS kernel presentation

Editor icons



Description

General function :

This tool gets a brightness threshold and a starting pixel as inputs. The brightness of the starting pixels should be below the threshold but there should be at least one pixel next to the starting pixel that has brightness above the threshold. The tool approximates the position between the dark and the bright pixels that would have brightness equal to the threshold. Then the tool seeks the next pixel on the dark side of the border and again approximates the position between that pixel and its bright neighbors that would have brightness equal to the threshold. And so on. If requested a selected number of additional points can be inserted between two of the full pixel results. These additional points reduce the difference in real length between steps in direction of the grid (to the left, right, up, down) and steps at 45 degrees to the pixel grid (up-left, up-right, down-left, down-right). The sub-pixel approximation of all the positions leads to a real flowing curve with an almost (but not fully) equal step width.

The contour following is done either in clockwise or in counterclockwise direction. It depends on the image if this distinction makes a difference and what direction is of better use.

The contour ends when it is closed (reaches the same point again). A special case of that condition is a dead end of the contour or a very small local loop that can artificially stop the contour by making a full loop within very few pixels. In such cases the image could be 'improved' by filtering or better illumination. The contour will stop also if it reaches the edge of the working area of the tool.

Note : All the sub-pixel approximations are done with an accuracy of 0.25 pixels for speed reasons.

Results :

The coordinates of all the pixels of the contour are written to the point-list buffer. The number of points is returned as a result of the tool.

Point-list item usage:

Item field	Object data
Point	The coordinates of the contour point

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Start Point	The coordinates of the start point
Center	Center point of the rectangular workspace
Width	The rectangle width in pixels
Height	The rectangle height in pixels
Angle	Rotation angle (currently ignored)
Direction	Clockwise / Counterclockwise contour following
Threshold	The threshold for the underlying binarization – brightness level on which the contour travels
Start position	Start index in the point-list where pixels found will be stored
Section size	Maximum contour length (to prevent point-list overfill)
Dash length	The rectangle line style: 0 = solid line, non-zero = dashed line
Sub-pixels	Shows the number of sub-pixels to insert between each pair of physical contour pixels. 0 = no sub-pixels, maximum : 10
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Contour length	Number of contour points found and written to the point-list

Similar tools

Edge detection 2	If you request a very high number of search lines (i.e. equal to the height of the workspace) you get a lot of pixels on a given edge and all of them are written to the point-list buffer. While in special cases they will appear there well ordered this will not be the case if the edge has a complex geometry. This tool could be an alternative in some cases if for instance you want to follow up with a 'Point-list best circle' where the sequence does not matter if the 'Contour' tool is not appropriate for some reason. An important consideration is to be given to the edge direction. The 'Contour' tool always approximates orthogonal to the edge while the 'Edge detection 2' tool approximates always in the direction of its search lines but does a better sub-pixel resolution. So normally the 'Contour' tool would be more precise but if the real direction of the edge is known the 'Edge detection 2' could be better (noisy images).

Usually combined with

3x3 filter	To improve the stability against sudden stops of the contour following generated by noise it may be necessary to filter the image with a low pass filter (smoothening).
Find blobs	In the 'Connected' mode a good starting point for a contour around it is generated for every object found. Please use the 'Get blob' tool for easy access.
Edge detection 2	Another method to get a valid starting point is to use the second group of results from this tool. Please read more in the description of that tool.
Compare point-lists	
Fast compare point-lists	These tools can be used to compare the actual contour with stored template contours and solve tasks like identifying the object, checking for any differences between ideal and real, finding the precise location of the object on a conveyor, checking for completeness and so on.
Contour matching	That is a very strong tool that allows comparing a set of actual contours with a stored set of template contours. The object all the contours belong to can be freely rotated. Good post-processing is done with the 'Match filter' tool.
Other point-list tools	for post-processing of the resulting contour pixels, i.e. 'Point-list best line/circle'

Tips & Tricks

Contours around an object can easily be used to find out, if the object is damaged, where it is positioned, at what rotation and so on. If the 'Find blobs' tool is used to find the center of an object, the 'Point-list distance' tool can then be used to find the distance between all of the contour points and that center. From there a 'Point-list sort' or 'Point-list filter' tool can be used to find the minimum and maximum distance or the number of points outside given thresholds. In many cases even just the length of the contour is a good indicator if all is right or not.

Do not forget to set a maximum contour length after you have an impression what would be the worst normal case in your application. If for some reason (dusty lenses after two years) the contour does not close but does three additional loops around the object many problems may arise especially if you work in a real-time environment. The contour following will take much longer, the post-processing will take much longer since it has to process all of the contour points, you may even overfill the point-list buffer and get an error which may hang the system. So better set a maximum length and after the tool finishes compare the number of points found. If it is equal to the maximum length you set then something is wrong and you should act accordingly.

Examples

Not indexed yet – sorry.

2.14. Image Area

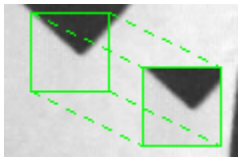
Group

Image-processing tools

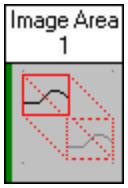
Short description

This tool provides a lot of different operations that use one or two rectangular areas. They can be located in the ‘Frame buffer’ (actual image) or the ‘Freeze buffer’ (secondary image). The possible operations include masking (maximum, minimum), moving/copying, shrinking (sub-sample, pyramid), inverting and combining (NOT, ADD, SUB, AND, OR, XOR) of rectangular areas.

VIMOS kernel presentation



Editor icons



Description

General function :

There are two full screen images directly accessible by VIMOS – the ‘Frame buffer’ and the ‘Freeze buffer’. The ‘Frame buffer’ is the normal actual image. The ‘Freeze buffer’ can be used as a temporary storage place for a full screen image or for smaller partial images. You can imagine it is positioned directly behind the ‘Frame buffer’ so that the overlay drawings mark the same spot in both buffers.

The ‘Image area’ tool allows to position two non-rotated rectangular workspaces and to execute an operation either from one workspace into the other or to combine the content of both areas into the second workspace. The user has the choice for both workspaces to define if the ‘Frame buffer’ or the ‘Freeze buffer’ should be accessed.

Note : While most tools are also executed the ‘Edit mode’ this tool works only in ‘Run mode’.

Supported operations :

Operation	Description
COPY	Copy the source area into the destination area
AND	Combine pixels of both areas using bitwise AND

OR	Combine pixels of both areas using bitwise OR
XOR	Combine pixels of both areas using bitwise XOR
NOT	Copy an inverted source area into the destination area
ADD	Sum of source and destination pixels without overrun
SUB	Absolute difference between source and destination pixels
MIN	Minimum brightness of source and destination pixels
MAX	Maximum brightness of source and destination pixels
PYRAMID	Shrink to half of the original size using interpolation (good, slower)
SUBSAMPLE	Shrink to some ratio of the original size by dropping pixels (worse, faster). Specify **Dest width** < **Source width** and **Dest height** <= **Source height**. **WARNING:** This operation may produce illegal results on the camera if the source and the destination rectangles overlap. If you can't get rid of such overlapping, place the destination rectangle above the source rectangle as much as possible.

Results :

There are no other results of this tool then the changes in the image.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Source pt	Center point of the source rectangle
Dest pt	Center point of the destination rectangle
Source width	Width of the source rectangle
Source height	Height of the source rectangle
Dest width	Width of the destination rectangle
Dest height	Height of the destination rectangle
Source page	Source image ('Frame buffer' or 'Freeze buffer')
Dest page	Destination image ('Frame buffer' or 'Freeze buffer')
Operation	Operation to apply on the image areas
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

This tool has no results.

Similar tools

Copy image	copies the whole content of the ‘Frame buffer’ to the ‘Freeze buffer’ or back. Since this could be done with the ‘Image area’ tool the older ‘Copy image’ is a bit outdated, but could have a better speed if the whole image has to be copied.
3x3 filter	is able to do very local operations between pixels. So the brightness of the left pixels could be subtracted from that of the right pixels within the matrix. That would detect fine vertical lines.

Usually combined with

Load image area	could be used before the ‘Image area’ tool to load (for instance) a masking image area into the ‘Freeze buffer’. That image area could then be used with the ‘AND’ operation to mask out parts of the destination. Another typical use would be to load an image that has the brightness distribution of the illumination on the blank background. By using the ‘SUB’ operation of the ‘Image area’ tool this (uneven) light distribution could be subtracted from the actual image so that an even illumination is simulated (shading correction).

Tips & Tricks

If you don’t need the whole image resolution but need the highest possible execution speed of your image processing it may be useful to use the ‘Pyramid’ or even the ‘Subsample’ function of this tool to reduce the image real estate to a quarter or even more. If you use ‘Pyramid’ you may even be making a low pass filtering with the ‘3x3 filter’ tool superfluous. The shrinking of the image improves the speed of filtering, contour following and others.

Another good use of shrinking is the size reduction of any images that have to be saved to the flash or to the ‘Freeze buffer’. You could save four images instead of one.

Examples

Not indexed yet – sorry.

2.15. Save Image Area

Group

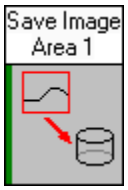
Image-processing tools

Short description

This tool saves a rectangular area of the image to file (flash memory).

VIMOS kernel presentation

Editor icons



Description

General function :

This tool saves a rectangular image area to file (flash memory). Later, the image area can be load back by using the ‘Load image area’ tool. The VIMOS-Simulator on the PC is able to upload and download such files to and from the camera flash memory (Camera Files Functions). On the PC they are just normal bitmap files, so that it’s easy to create and simulate first on the PC and then upload the result to the camera.

Note : While most tools are also executed in the ‘Edit mode’ this tool works only in ‘Run mode’.

Warning : This tool writes to file, which in case of flash memory could quickly consume all available space ! In that case you would have to leave VIMOS and use the ‘Camera Files Functions’ of the Simulator to delete some of the files (always include the last because it will be truncated) and then call a ‘Pack Flash’ operation to regain free flash memory. It’s important to know that you will not be able to save your program and setup when leaving VIMOS if the flash is full. So better don’t use this tool in ‘Run-mode’ without ‘hands on’ and by all means please save your work BEFORE you first start such a program. **Even if you turn off the auto-incrementing so that you always overwrite an older file please be aware, that this will not regain the old space of the flash memory, so it will fill up.** It may be different for other media like MMC or SD flash cards.

Results :

The tool has an ‘auto incrementing’ feature to generate a sequence of filenames. It returns the number that was currently used.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Center point of the rectangular workspace
Width	Rectangle width in pixels
Height	Rectangle height in pixels
Source	'Frame buffer' or 'Freeze buffer'
File prefix	First part of the filename
File #	Second part of the filename – number, can be auto-incremented
Auto-increment	[On/Off] – Off does not help against flash fill-up !!
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
File #	Actual suffix of the filename

Similar tools

Save point-list

Save string buffer — are other tools that write to flash. You could use the 'Point-list read pixels' tool to read the brightness values of certain pixels into the point-list and then save the point-list. But since one single item of the point-list is much bigger then a pixel of an image area you'll save space only with short point-lists.

Send image area — could be used to transfer the image area via serial interface to a host PC where supporting tools exist that could write it automatically to a (big) hard disk.

Usually combined with

Correlation Init — needs a template within the 'Frame buffer' (actual image). To save and restore that template 'Save image area' and 'Load image area' could be used.

Tips & Tricks

Just don't forget to think about the free space in the flash (see above). This tool needs some time if it is writing to flash memory. So if you execute it to save error images please make sure you have enough time if you are working in a real-time environment. You could consider to make a temporary copy to the 'Freeze buffer' by using the 'Image area' tool and to save from there at the right moment.

Examples

Not indexed yet – sorry.

2.16. Load Image Area

Group

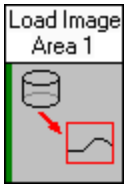
Image-processing tools

Short description

This tool loads a rectangular area of the image from file (flash memory).

VIMOS kernel presentation

Editor icons



Description

General function :

This tool loads a rectangular image area from file (flash memory) and displays it at a given position in the ‘Frame buffer’ (actual image) or puts it to the ‘Freeze buffer’ (secondary image). Such an image area file could have been created with the ‘Save image area’ tool. The VIMOS-Simulator on the PC is able to upload and download such files to and from the camera flash memory (Camera Files Functions). On the PC they are just normal bitmap files, so that it’s easy to create and simulate first on the PC and then upload the result to the camera.

Notes:

- While most tools are also executed in the ‘Edit mode’ this tool works only in ‘Run mode’.
- The image area should be positioned fully within the screen boundaries. Since the position is given only by a point that marks the center it depends on the size of the image area that was stored to the loaded file where you could position the center point.

Results :

The tool has an ‘auto incrementing’ feature to access a sequence of filenames. It returns the number that was currently used.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Center point of the loaded image area
Destination	‘Frame buffer’ or ‘Freeze buffer’
File prefix	First part of the filename
File #	Second part of the filename – number, can be auto-incremented
Auto-increment	[On/Off]
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
File #	Actual suffix of the filename

Similar tools

Load point-list

Load string buffer are other tools that read from flash.

Usually combined with

Correlation Init needs a template within the ‘Frame buffer’ (actual image). To save and restore that template ‘Save image area’ and ‘Load image area’ could be used.

Tips & Tricks

Instead of loading an image area during every program cycle from flash it would be faster to load it only during the first cycle and then use the ‘Image area’ tool to copy it from the ‘Frame buffer’ (actual image) to the ‘Freeze buffer’ (secondary image).

Examples

Not indexed yet – sorry.

2.17. Correlation Init

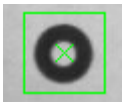
Group

Image-processing tools

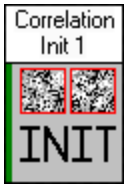
Short description

This tool is used to initialize (learn) a pattern image for the ‘Correlation exec’ tool.

VIMOS kernel presentation



Editor icons



Description

General function :

Correlation is an operation that finds those places in an image that have differences less then a given threshold to a given (smaller) template image. The ‘Correlation exec’ tool does this by using a template out of a special buffer that can hold a certain amount of such templates (depending on camera model and template size). This tool transfers a rectangular image area into that buffer.

Results :

The tool returns a number (index) that identifies the template. To run the ‘Correlation exec’ tool with this pattern you need to give it that index. The number is incremented every time the ‘Correlation Init’ tool is called until VIMOS is restarted or the ‘Free pattern’ tool is called. Just exiting ‘Run mode’ and starting it again does NOT restart the indexing and does NOT free the special buffer so you can fill the buffer with one program and the start another using the templates. On the other hand it is necessary to put a ‘Free pattern’ tool in the initialization part of the program before starting to call ‘Correlation init’ to make sure the buffer is empty. Since it is tricky to be sure about the index actually assigned to a template it is better to make a link between the ‘Correlation init’ and the ‘Correlation exec’. If that can not be done directly because you want to use one ‘Correlation exec’ with different templates, it is possible to write all their indexes to element parameters of the point-list buffer by using the ‘Set point-list parameter’ tool, which later can be recalled with a ‘Get point-list parameter’ tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Center point of the rectangular area that is saved
Length	Rectangle length in pixels
Height	Rectangle height in pixels
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Pattern Id	Index (identifier) of learned pattern image

Similar tools

Object Recognition Init initializes the template for the edge based object recognition. That operation is rotation and size independent and works even with partially hidden objects.

Usually combined with

Load image area to load pattern images from permanent memory (flash) into the actual image from where you can use ‘Correlation Init’ to transfer them to the pattern buffer

Correlation exec tool to perform the correlation with the pattern

Free pattern tool to delete the pattern and free up the memory used by it – use this before starting to fill the pattern buffer

Tips & Tricks

Please read the description chapter above about how to use this tool. It should work fine if you use ‘Free pattern’ at initialization and check for errors while filling the pattern buffer.

Please read the information in the following part about the ‘Correlation exec’ tool. For instance there is a limitation for the maximum pattern size in some of its operation modes.

Examples

Not indexed yet – sorry.

2.18. Correlation Exec

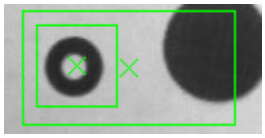
Group

Image-processing tools

Short description

The tool is used to execute a correlation operation using one of the template image areas previously generated by the ‘Correlation Init’ tool.

VIMOS kernel presentation



Editor icons

Description

General function :

Correlation is an operation that finds those places in an image that have differences less then a given threshold to a given (smaller) template image. The ‘Correlation exec’ tool does this by using a template out of a special buffer that can hold a certain amount of such templates (depending on camera model and template size). The ‘Correlation Init’ tool transfers a rectangular image area into that buffer.

The ‘Correlation exec’ tool performs a normalized grayscale correlation with eliminated mean values. That means that the tool does the comparison between template and actual image by looking on RELATIVE brightness changes rather then by comparing directly the absolute values. That way the tool copes with brightness changes that make the whole image brighter or darker.

There are different search modes (fine, normal, fast). The faster the mode the more pixels are skipped. Since the finer modes use more memory because they look on every pixel their maximum template size is restricted to about 32x32 for the fine mode and about 152x152 for the normal mode (depends on camera, so please check).

Results :

The results of the pattern-matching process are recognition rates in range [0,1024], and pattern coordinates (locations) which are saved in the point-list buffer. Each item in the result point-list contains the following parameters:

Point	Center point of found pattern

Param 0	Recognition rate in the range [0,1024]
Param 1	Pattern width in pixels
Param 2	Pattern height in pixels

Rates above 870 usually indicate the presence of the pattern, but it depends on the application. In noisy or distorted images the value may be smaller, but then the risk of wrong positives rises. This tool needs to be checked out with the real images from the application. It returns the number of locations found.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Center point of the rectangle workspace
Angle	Rectangle rotation – DO NOT USE, only 0 degrees supported – sorry
Width	Rectangle width in pixels
Height	Rectangle height in pixels
Threshold rate	[700, 1024] – no location with match value below is recognized
Pattern id	Index (identifier) of pattern image, stored by the 'Correlation Init' tool. Link this argument to the result of a 'Correlation Init' tool.
Pt-list start	Start position in the point-list where the results are stored
Max find	The maximum number of returned results
Search mode	Fine, Normal, Fast (different pixel skipping rates)
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Found	Tells how many positions where found (<= Max find)

Similar tools

Object recognition exec executes the edge based object recognition. That operation is rotation and size independent and works even with partially hidden objects.

Usually combined with

Correlation Init tool to initialize the pattern(s).

Free pattern tool	to delete the pattern and free up the memory used by it – use this before starting to fill the pattern buffer

Tips & Tricks

In general, larger patterns are recognized more reliably, especially in coarse (fast) searching mode. It is essential for the successful pattern recognition to use patterns with not too big a difference between the amounts of bright and dark pixels.

To get stable results in x and y direction you need patterns that have a cross or at least Y like structure – or let's say a small rectangle or circle. If you have just a line within your pattern and in the image is a long line at the same angle, then you'll get a lot of results along that line (depending on the 'Max find' argument).

Examples

Not indexed yet – sorry.

2.19. Object Recognition Init

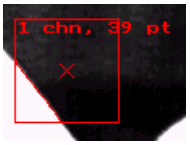
Group

Image-processing tools

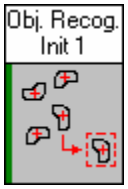
Short description

The tool learns an object (initialization of an edge-pattern) the “Object Recognition Exec” tool should search next.

VIMOS kernel presentation



Editor icons



Description

General function :

The tool learns an object from input image (called *pattern image*) and saves formalized description of the object into kernel memory buffer. The tool learns an image in a user-defined rectangle. The pattern image should not be smaller than 12x12 pixels. The “Object Recognition Exec” tool uses this description data to find the object in a search image. The connection between the two tools is done by linking the result of the “Object Recognition Init” tool to the “**OR init result num**” argument of an “Object Recognition Exec” tool. There is no need to execute the “init” tool each time before the “exec” tool. We recommend performing the learn-operation once – for example in cycle 0 (the first cycle of the main system loop after starting the user-program).

It is important to adjust input arguments "**Min edge height**" and "**Min edge grad**" (together with light and focus) to achieve clear and stable detection of edges and to remove noise. Use "**Min chain len**" argument to remove redundant short contours. Best learning results are obtained when edge contours are longer than 24 pixels.

Note:

The argument “**Show edges**” is intended to help finding better image edges, but it will destroy the source image. Currently this option is disabled. Use the “Object Recognition Exec” tool in destructive mode for that purpose.

Results :

The tool returns one float result – an identifier of learned object. This result must be linked to the 'Object Recognition Exec' tool. This result is a kind of pointer to buffer, allocated in DRAM. The tool generates object description data, which is stored in this buffer. This data can't be saved in a flash file yet, so every time you want to use the 'Object Recognition Exec' tool, you have to execute the 'init' tool before. Currently due to DRAM limitations you can execute this tool only once in the user program. If you start more than one pair of 'Object Recognition Init' and 'Object Recognition Exec' tools you will receive unexpected results.

Algorithm

The tool is supporting all processing steps:

- edge detection
- contour tracking
- basic pattern point extraction

Arguments

Argument	Description
Point	Center point of learn-image.
Width	Width of learn-image in pixels.
Height	Height of learn-image in pixels.
Min edge height	Minimum edge height used in edge-detection (7 to 127). Valid edges are borders between light and dark image areas with pixel difference above the argument value.
Min edge grad	Minimum edge gradient used in edge-detection (from 6 to "**Min edge height**"). The edge gradient represents the edge sharpness.
Min chain length	Minimum length of contour chains in pixels.
Show edges	Specifies mode of tool operation: • **Off**. Normal learn mode • **On**. Destructive mode (currently **disabled**). The tool stores the edge image into the frame buffer overwriting the input learn-image. No further processing is done. This option is useful in the tuning phase of the tool, when tool arguments are adjusted to achieve clear and stable detection of object contours.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Pattern Id	Identifier of learned object (pointer for "Object Recognition Exec" tool).

Similar tools

Correlation Init tool	to initialize the pattern(s) for Correlation Exec tool

Usually combined with

Load image area	to load pattern images from file into actual image from where you can use ‘Object Recognition Init’ to initialize pattern object.
Object Recognition Exec	to find the learned object

Tips & Tricks

If you need an image with black background and edges in white color you can use this tool in destructive mode without ‘Object Recognition Exec’ tool (destructive mode would be available in future VIMOS versions).

Examples

Not indexed yet – sorry.

2.20. Object Recognition Exec

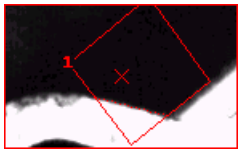
Group

Image-processing tools

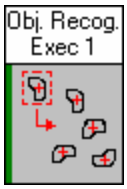
Short description

The tool finds rotated, scaled and possibly occluded object(s) previously learned by the “Object Recognition Init” tool.

VIMOS kernel presentation



Editor icons



Description

General function :

The tool performs 2-D object recognition. The object must be learned from a pattern image by the “Object Recognition Init” tool. The tool can search the learned object multiple times in different search images. The minimum size of the search image is 32x32 pixels.

An exemplary tool execution sequence is:

- Learn pattern image on cycle 0 of the user-program by the “Object Recognition Init” tool. Link “Object Recognition Init” result to “**OR init result num**” argument of an “Object Recognition Exec” tool (should be executed after the “init” tool). Skip “Object Recognition Init” execution in cycles above 0.
- Take pictures (automatically or by the “Take picture” tool) and execute the “Object Recognition Exec” tool (all system cycles). Recognition results are stored in the point-list buffer.

To achieve good recognition results the tool needs fine-tuning of input arguments. The edge detection arguments "**Min edge height**" and "**Min edge grad**", along with appropriate lighting, should be adjusted to achieve clear and stable detection of meaningful contours, as well as to remove noise and clutter. The "**Min chain len**" argument should be adjusted to remove redundant short contours. Best recognition results are obtained when edge contours are longer than 24 pixels.

The number of object recognition results is limited by the value of "**Max items search**". This parameter limits also the searching phase in “**Restric. search**” = “Yes” mode. This mode should only be used after achieving stable recognition results when restricted search is disabled (“**Restric. search**” = “No”). In restricted search mode the tool stops processing when first "**Max items search**"

results with recognition rate above "**Min rate**" are detected. Turn off restricted search to find best "**Max items search**" results.

All argument values are presented either in their natural units (e.g. rotation 90 degrees), or in units of 1/1024th (e.g. rate 1024 == 100%, scale 1126 == 110%).

In case of recognition fail, decreasing of "**Min rate**" and/or increasing of "**Contour width**" may help. Internally the tool works with odd values of the second argument. In general, "**Contour width**" should have small values in case of small/fine patterns and when we try to achieve more precise recognition rates.

To achieve better execution times the scale range ["**Min scale**", "**Max scale**"] and "**Rot sector width**" (size of rotation angle range) should be kept as narrow as possible. The "**Rot sector width**" argument should be used together with "**Rot sector start**", which specifies beginning angle of rotation range (sector) in clockwise direction (positive values only).

You can enable the option "**Show edges**" to find better pattern edges, but this will destroy the source image (searching is disabled). Use this option only tuning of tool's arguments.

The "**OR init result num**" is a kind of pointer, which must be linked only to result of an "Object Recognition Init" tool and should not be set manually. Currently you should not execute more than one pair of 'Object Recognition Init' and 'Object Recognition Exec' tools otherwise you will receive unexpected results.

Performance considerations:

In case of good recognition results but bad tool speed, you may reduce execution time by setting "**Restric. search**" to "Yes". In this mode both detection ability and searching time are proportional to "**Max items search**" and depend heavily on "**Min rate**" and "**Contour width**"). Set "**Min rate**" about 10% below the estimated rate. Set "**Max items search**" argument to be 2-10 times greater than the number of real objects in the search image. After achieving stable recognition results in relatively short searching time you may check system behavior slightly varying "**Contour width**" (and possibly "**Min rate**").

Remember that more edges in the search image increase the execution time.

Results:

Recognition results (parameters of detected objects) are stored in the point-list buffer, starting from offset "**Res ptlist start**". The results are sorted in descending order in respect to the recognition rate. Each point-list item contains result parameters in the following item fields:

Point-list item parameter	Recognition results
Point	Coordinates of rotation and scale origin point relative to the upper left corner of the screen. The rotation/scale origin point is the top-left corner of the rotated and scaled rectangle, which encloses the found object.
Param 0	Recognition rate (above the user-defined threshold "**Min rate**").
Param 1	Rotation angle (in the user-defined range of allowed angles).
Param 2	Scale value (in the user-defined range of allowed scale values).
Param 3	X-coordinate of the rotation/scale origin point in the learn image. Param 3 and Param 4 show the position of the origin point relative to the learn image. This origin point is the top-left corner of a non-rotated and non-scaled rectangle, relative to the learn image, which corresponds to the rotated rectangle found by the tool (see the "**Point**" parameter).
Param 4	Y-coordinate of the rotation/scale origin point in the learn image.

The tool returns one float result - number of detected objects with rate above "**Min rate**" (less than or equal to "**Max items search**")

Algorithm

The tool algorithm includes the following principal processing steps:

- Edge detection.
- Contour tracking.
- Basic pattern point extraction.
- Pattern recognition.

Arguments

Argument	Description
Point	Center point of search image.
Width	Width of search image in pixels.
Height	Height of search image in pixels.
Min edge height	Minimum edge height used in edge-detection (7 to 127). The edge height represents pixel difference between neighboring image areas with different pixel intensities (darker and lighter).
Min edge grad	Minimum edge gradient used in edge-detection (from 6 to "**Min edge height**"). The edge gradient represents the edge sharpness (or edge width in number of pixels).
Min chain len	Minimum length of contour chains in pixels (4 to 24 pixels).
Restric. search	Specifies restricted search mode: No = disable, Yes = enable
Max items search	Maximum number of searched objects.
Min rate	Minimum recognition rate (160 to 1023, 1024==100%). Detected objects with lower rates are not included in the result list.
Contour width	Minimum contour width: 3 to 9 pixels, internally set to odd value.
Min scale	Minimum scale value, 1024==100%, keep less than or equal to "**Max scale**".
Max scale	Maximum scale value, 1024==100%, keep less than 200%.
Rot sector start	Start angle of allowable range (sector) of rotation angles (-180 to 179 degrees).
Rot sector width	Size of allowable range (sector) of rotation angles (0 to 360 degrees).
Pattern Id	Identifies the pattern image, which must be searched by the tool. This argument should be linked to a result of "Object Recognition Init" tool.
Res ptlist start	Start position in the point-list buffer, where a list with recognized objects is stored.
Show edges	Specifies mode of tool operation: • **No**. Normal recognition mode • **Yes(destructive)**. The tool stores the edge image into the frame buffer overwriting the input search image. No further processing is done. This option is useful in the tuning phase of the tool, when tool arguments are adjusted to achieve

	clear and stable detection of object contours.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Number of objects	Number of detected objects with recognition rate above "**Min rate**" (<= "**Max items search**").

Similar tools

Correlation exec tool to perform the correlation with the pattern learned with Correlation init tool

Usually combined with

Object Recognition Init initializes template for edge based object recognition. The operation is rotation and size independent and works even with partially hidden objects.

Tips & Tricks

Examples

Not indexed yet – sorry.

2.21. Median filter

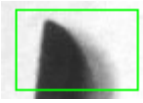
Group

Image-processing tools

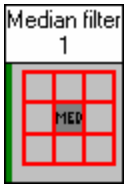
Short description

The operator applies a median filter on a region of the image defined by a rectangle.

VIMOS kernel presentation



Editor icons



Description

General function:

The tool applies a median filter on a region of the image defined by a rectangle. The arguments **Point**, **Width**, and **Height** define the rectangular workspace. The argument **Type** specifies the size of the filter window.

Results :

This tool has no results.

Algorithm

The tool performs median filtering by a sliding window over pixels in a rectangle. The median pixel of the input window will replace the center pixel of output window. As a result of this every pixel with distinct intensity will be like its neighbor's intensities, i.e. the median filter eliminating intensity spikes. The size of window must be odd. The tool uses tree window types – 3x3, 5x5, 7x7. The tool uses a sort algorithm for fast median searching.

Arguments

Argument	Description
Point	Center point of the rectangular workspace
Width	Rectangle width in pixels

Height	Rectangle height in pixels
Type	Size of filter window – 3x3, 5x5, 7x7
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

This tool has no results.

Similar tools

Median Filter point-list — Applies a median filter (low pass, softening) to a sequence of values located within a given parameter of a part of the point-list buffer. A standard way to use this would be first to fill the points of the elements of that part of the point-list with coordinates of pixels (i.e. by calling the 'Generate point-list circle' tool). Then use the 'Point-list read pixels' tool to get the actual brightness into one of the parameters of these point-list elements and finally call 'Filter point-list'. After that i.e. the 'Point-list Edge detection' tool can be used ...

Usually combined with

This tool is normally used for filtration before another tool from "Image processing" group as Fast edge detection, Edge detection 2, Find blobs, Contour, Barcode, OCR and others.

Tips & Tricks

This tool is not one of the fastest because it requires a lot of memory access and calculations. It is possible to use the 'Pyramid' operation of the 'Image area' tool to reduce the size of the area that needs to be filtered. Since this operation has a noise reducing effect it may even improve the image enough to make the use of the 3x3 filter unnecessary.

If you need to preserve an area you need to filter you can use the 'Copy' operation of the 'Image area' tool and copy an area to another place or to the Freeze buffer. After you finished with the filtering and the following tasks you would be able to copy the original back or to process it at its new place.

Examples

Not indexed yet – sorry.

2.22. Watershed segmentation

Group

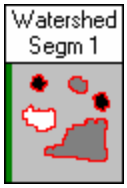
Image processing tools.

Short description

This tool performs region segmentation of a gray-scale image.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool finds regions with approximately uniform pixel intensity in a gray-scale image and generates output image, where different regions are filled with different uniform gray-scale colors. There is option which draws 1-pixel wide boundary line between regions with pixel values 255 (white). The tool can be used to remove light distortions when light color in one image area is darker then dark color in other image area. This is necessary for some tools like OCR, which work with fixed binarization threshold in the whole image. The tool can be use also to filter image noise and to recover corrupted areas in black-and-white images by region growing.

Results:

The tool overwrites output image with colored regions over the input image (in current frame buffer). The tool has 1 float result – the number of detected regions.

Algorithm

The tool works in several stages. On each stage an intermediate image overwrites the input image for the stage. Here are the basic processing stages:

- The image is blurred with blur iterations, specified by the argument “Blur. iterations”.
- Region contours are generated by a Sobel filter with gradient. The contour-detection sensitivity is controlled by the argument “Min. gradient”. Greater gradient values require better region contours (more distinct edges between regions).

- Contour thinning and generation of ridge image – optional, controlled by the argument “Use ridges”.
- Negative distance transform, controlled by “Ndist. iterations” argument (0: skip this stage). This transform should be used in case of bad region borders.
- Final stage with region segmentation and boundary generation. The argument “Boundaries” specifies boundary width in pixels (0:none, 1: draw region boundary). The first option can be used with or without negative distance transform. The second option should be used when negative distance transform has been enabled and the tool does not produce correct results due to asymmetrical region boundaries, generated by the negative distance transform.

The Sobel and the negative distance intermediate images can be passed as final result image - see argument “Output mage”.

Arguments

Argument	Description
Point	Center point of image rectangle.
Width	Image width in pixels.
Height	Image height in pixels.
Blur. Iterations	Number of blurring iterations needed to generate blurred image.
Min. gradient	Contour detection sensitivity. Lower gradient values specify higher sensitivity. Higher gradient values specify lower sensitivity but more noise-tolerant processing.
Use ridges	Specifies contour thinning after generation of region contours by a Sobel filter.
Ndist. Iterations	Specifies negative distance transform with N iterations (N=0: disable).
Boundaries	Specifies generation of 1-pixel wide region boundaries: • **0-pixel parts, Symm**. Symmetrical boundaries, no region border lines. • **1-pixel width, Asymm**. Asymmetrical boundaries after negative distance transform. Generates 1-pixel wide region boundaries with color 255. Use this option after negative distance transform when you receive bad result image.
Output image	Specifies result image: • **Regions.** Final result image when all algorithm stages have been performed. • **Gradient (ridges).** The result image is the output image of the Sobel stage. If ridges are enabled, the result image is the ridge image. • **Negative distance.** The result is image is the output image of the negative distance transform (if enabled by non-zero number of iterations).
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Number of regions	Number of detected regions.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

2.23. Color Pixel Counter

Group

Image-processing tools

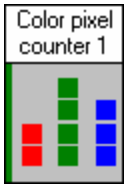
Short description

This tool counts color pixels in an image, which are in a given range.

VIMOS kernel presentation

Note: This is tool's presentation in Simulator and on cameras with video-output. On VCM40 sensor-camera (no video-output) the tool has no graphical representation.

Editor icons



Description

General function:

The tool calculates the number of color pixels, which are within specified color range. Min and max values for each component (R, G1, G2 and B) of color pixels define the color range. Three different categories (A, B and C) may be checked, each with separate set of min/max values.

Each color pixel is coded in the input image by a 2x2-pixel matrix in two sequential image rows and columns:

	Odd column	**Even column**
Odd row	**G1** (green 1)	**R** (red)
Even row	**B** (blue)	**G2** (green 2)

The input color image should be in Bayer matrix color format:

```
G R G R G R G . . .
B G B G B G B . . .
G R G R G R G . . .
B G B G B G B . . .
. . . . . . . . . .
```

The input image is read from the VIMOS frame buffer. It should be compatible with image format of the "Take CMOS Image" tool. The upper left corner of the input image rectangle is aligned to odd row and column (assuming row and column indexes begin from 1).

Faster processing is done when odd or even image rows are ignored, or when vertical and/or horizontal step is greater than 1.

The argument **Binar** specifies destructive mode of operation, in which the input image is binarized according to the tool's results:

- Good color pixels are marked by storing pixel values 0x80/0x40/0x20 for A/B/C categories respectively when selected by **Binar**. The binarization is performed for categories enabled by **Check cat** only.
- Bad color pixels are marked by 0x00.
- Skipped pixels (horizontal or vertical step > 1, odd or even lines ignored) are left unchanged.

Results:

The tool returns 8 float results – pixel counts and good/bad flags for each category A,B,C; and total tool results – index of best good category and flag for present good category.

Arguments

Argument	Description
Point	Center point of input image rectangle (compatible with "Take CMOS Image" tool).
Binar	Specifies destructive mode of operation: • **None** – disables destructive mode. • **A, B, AB, C, AC, BC** or **ABC** – enables destructive mode. The tool sets good and bad color pixels for specified categories in the input image (see description above).
Width	Width of input image.
Height	Height of input image.
Check cat.	Select 1-3 categories for calculation of color pixel counts with different min/max values. • **Cat. A**. Select category A. • **Cat. AB**. Select categories A and B. • **Cat. ABC**. Select categories A, B and C.
Odd lines	Ignore or process odd sensor rows.
Even lines	Ignore or process even sensor rows.
Horiz. step	Horizontal step for color pixels (>=1). Step 1 means processing of sequential color pixels. Remember that each color pixel is defined by 2x2 matrix of image pixels. Step 2 skips one color pixel (2 image pixels), step 3 skips 2 color pixels (4 image pixels) and so on.
Vert. step	Vertical step for color pixels (>=1). Step 1 means processing of sequential pairs of image rows. Remember that each color pixel is defined by 2x2 matrix of image pixels. Step 2 skips one color pixel row (2 image rows), step 3 skips 2 color pixel rows (4 image rows) and so on.
A: Oomin	Category A: Odd row, odd column (**G1** min value [0,255]
A: OOmax	Category A: Odd row, odd column (**G1**) max value [0,255]
A: OEmin	Category A: Odd row, even column (**R**) min value [0,255]

A: OEmax	Category A: Odd row, even column (**R**) max value [0,255]
A: EOmin	Category A: Even row, odd column (**B**) min value [0,255]
A: EOmax	Category A: Even row, odd column (**B**) max value [0,255]
A: EEmin	Category A: Even row, even column (**G2**) min value [0,255]
A: EEmax	Category A: Even row, even column (**G2**) max value [0,255]
A: Min.Cnt	Category A: Min color pixel count for good result.
A: Max.Cnt	Category A: Max color pixel count for good result. The category receives good result if the calculated pixel count is in the range [Min.Cnt, Max.Cnt].
B: Oomin	Category B: Odd row, odd column (**G1** min value [0,255]
B: Oomax	Category B: Odd row, odd column (**G1**) max value [0,255]
B: Oemin	Category B: Odd row, even column (**R**) min value [0,255]
B: Oemax	Category B: Odd row, even column (**R**) max value [0,255]
B: Eomin	Category B: Even row, odd column (**B**) min value [0,255]
B: Eomax	Category B: Even row, odd column (**B**) max value [0,255]
B: Eemin	Category B: Even row, even column (**G2**) min value [0,255]
B: Eemax	Category B: Even row, even column (**G2**) max value [0,255]
B: Min.Cnt	Category B: Min color pixel count for good result.
B: Max.Cnt	Category B: Max color pixel count for good result. The category receives good result if the calculated pixel count is in the range [Min.Cnt, Max.Cnt].
C: Oomin	Category C: Odd row, odd column (**G1** min value [0,255]
C: Oomax	Category C: Odd row, odd column (**G1**) max value [0,255]
C: Oemin	Category C: Odd row, even column (**R**) min value [0,255]
C: Oemax	Category C: Odd row, even column (**R**) max value [0,255]
C: Eomin	Category C: Even row, odd column (**B**) min value [0,255]
C: Eomax	Category C: Even row, odd column (**B**) max value [0,255]
C: Eemin	Category C: Even row, even column (**G2**) min value [0,255]
C: Eemax	Category C: Even row, even column (**G2**) max value [0,255]
C: Min.Cnt	Category C: Min color pixel count for good result.
C: Max.Cnt	Category C: Max color pixel count for good result. The category receives good result if the calculated pixel count is in the range [Min.Cnt, Max.Cnt].
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Pixel count A	Category A: Number of detected color pixels in specified range of min/max values.
Good A	Category A: Good result flag: • 0 = Bad result. • 1 = Good result: A: Min.Cnt <= Pixel count A <= A: Max.Cnt
Pixel count B	Category B: Number of detected color pixels in specified range of min/max values.
Good B	Category B: Good result flag: • 0 = Bad result. • 1 = Good result: B: Min.Cnt <= Pixel count B <= B: Max.Cnt
Pixel count C	Category C: Number of detected color pixels in specified range of min/max values.
Good C	Category C: Good result flag: • 0 = Bad result. • 1 = Good result: C: Min.Cnt <= Pixel count C <= C: Max.Cnt
Best category	Index of best good category - 0:A, 1:B, 2:C with highest pixel count. Equal to –1 if no good category detected.
Good	Flag for detected good category – 0:no, 1:yes.

Similar tools

Usually combined with

“Take CMOS Image” tool.

Tips & Tricks

Examples

Not indexed yet – sorry.

2.24. Color Conversion

Group

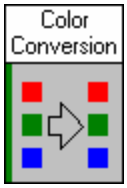
Image-processing tools

Short description

The tool converts color pixels from RGB to HSV format.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool converts color pixels in a rectangle image area from RGB format (red. green. blue) to HSV format (hue, saturation, value).

The input RGB color image should be in Bayer matrix format:

```
G R G R G R . . .
B G B G B G . . .
G R G R G R . . .
B G B G B G . . .
. . . . . . . . .
```

A 2x2-pixel matrix, located on two sequential image rows and columns in the input image represents one color pixel in RGB format:

G1 (green 1)	**R** (red)
B (blue)	**G2** (green 2)

The output color image is stored over the input image in HSV format:

```
V H V H V H . . .
S x S x S x . . .
V H V H V H . . .
S x S x S x . . .
```

. . . . . .

A 2x2-pixel matrix, located on each two sequential image rows and columns in the output image represents one color pixel in HSV format:

V (value)	**H** (hue)
S (saturation)	**x** (low-order bits)

The hue value **H** specifies the "color" of the pixel in the range [0,3600] (accuracy 0.1). The saturation value **S** specifies the purity of the color in the range [0,1000] (accuracy 0.001). Purity means how much black, white or gray is mixed with the color. **V** specifies brightness (intensity) of the color in the range [0,1000] (accuracy 0.001). The **x** byte contains the low order bits of the 3 values in the following format:

x = H H H H V V S S

where:

H H H H = 4 least significant bits of **H** value

V V = 2 least significant bits of **V** value

S S = 2 least significant bits of **S** value

The **x** byte is calculated when 'Precision' is on; otherwise it is set to 0.

The work area of the tool is a rectangle, specified by a center point, width and height. The rectangle dimensions should be even. When 'Horiz. step' and 'Vert. step' are equal to 1, the actual number of converted color pixels is ¼ of the total rectangle pixels.

The input image is read from the VIMOS frame buffer. It should be compatible with image format of the "Take CMOS Image" tool. The upper left corner of the input rectangle is aligned to odd row and odd column (assuming row and column indexes begin from 1).

***ATTENTION**. This notation is specific for this tool and the "Color Pixel Counter" tool, where rows and columns begin from index 1.*

.

The 'Mode" arguments specifies conversion mode – RGB to HSV or vice versa. The 'Precision' argument is ignored in HSV to RGB conversion and the tool always reads least-significant bits from the **x** byte.

Results:

The tool returns 1 float result – conversion error code.

Arguments

Argument	Description
Point	Center point of rectangle, which specifies the work area of the tool.
Width	Rectangle width in pixels (should be even).
Height	Rectangle height in pixels (should be even).
Precision	Conversion precision for RGB to HSV: • **Off**. Don't fill low order bits of HSV values into **x** byte (**x** is set to 0). • **On**. Fill low order bits of HSV values into **x** byte.
Horiz. step	Horizontal step for color pixels (>=1). Step 1 means processing of sequential pairs of image columns. Remember that each color pixel is defined by 2x2 matrix of image pixels. Step 2 skips one color pixel column (2 image columns), step 3 skips 2 color pixels

	columns (4 image columns) and so on.
Vert. step	Vertical step for color pixels (>=1). Step 1 means processing of sequential pairs of image rows. Remember that each color pixel is defined by 2x2 matrix of image pixels. Step 2 skips one color pixel row (2 image rows), step 3 skips 2 color pixel rows (4 image rows) and so on.
Mode	Specifies conversion mode: RGB to HSV or HSV to RGB.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Error code	Conversion error code (0=OK).

Similar tools

Usually combined with

“Color Pixel Counter”, which can be used to count pixels in a given color range regardless of pixel intensities. Convert RGB image to HSV format by this tool and then call the “Color Pixel Counter” tool.

Tips & Tricks

Use ‘Precision” = Off in the “Color Conversion” tool because currently you are not able to count full-precision HSV color pixels by the “Color Pixel Counter” tool.

Examples

Not indexed yet – sorry.

2.25. Erosion/dilation

Group

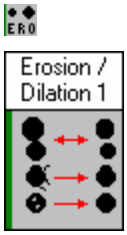
Image-processing tools

Short description

The tool performs erosion, dilation, opening and closing operations on a rectangle image area.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool works in rectangle image area defined by center point and rectangle width and height arguments. The tool first performs image binarization, which transforms input gray-level image to binary image. Gray-level pixels, which are in the range [‘Down threshold’, ‘Up threshold’] are treated as object pixels. Gray-level pixels outside the range are treated as background pixels.

The tool performs one of the following morphological operations over the binary image:

- **Erosion**. The tool performs ‘Steps’ erosions. Each erosion shrinks objects by 1 pixel. It removes object boundary pixels and enlarges inner object holes. Objects, which are smaller than the structuring element, disappear (see description of structural elements below). Use this operation to remove “salt and pepper” noise outside meaningful objects.
- **Dilation**. The tool performs ‘Steps’ dilations. Each dilation expands objects by 1 pixel. It adds pixels at object boundaries and fills inner object holes. Use this operation to delete “salt and pepper” noise inside meaningful objects (for example to fill small object holes).
- **Opening**. The tool performs ‘Steps’ erosions followed by ‘Steps’ dilations. Deletes noise and thin lines, which connect bigger objects, and preserves object sizes.
- **Closing**. The tool performs ‘Steps’ dilations followed by ‘Steps’ erosions. Fills small holes in objects and preserves object sizes.

The tool applies the following structural elements for erosion/dilation:

- **3x3 diamond**. This element shrinks or expands 4-connected pixels:

  ```
  0 1 0
  1 1 1
  0 1 0
  ```

- **3x3 box**. This element shrinks or expands 8-connected pixels:

  1 1 1

  1 1 1

  1 1 1

- **3x3 horizontal**. This element shrinks or expands objects in horizontal direction:

  0 0 0

  1 0 1

  0 0 0

- **3x3 diamond**. This element shrinks or expands objects in vertical direction:

  0 1 0

  0 0 0

  0 1 0

Finally the tool modifies the result image for better view and eventual post-processing by the “Find blobs” tool – it fills object pixels with ‘Object color’ and background pixels with ‘Bgnd color’.

***ATTENTION**. When running this tool in Simulator, use ‘Object color” and ‘Bgnd color’ values, which are multiple of 8. Set 16-bit or true 24-bit colors on your computer.*

.

Results:

The tool returns 1 float result – conversion error code.

Algorithm

The structuring element is placed at all possible positions inside the image rectangle. For each position of the element the tool does:

- Erosion. If the ON (non-zero) pixels of the structuring element are not completely overlapped by ON object pixels, then the image pixel at the element ***origin*** (the center pixel of the element) is set to OFF, i.e. it becomes a background pixel.
- Dilation. If some ON pixel of the structuring element overlaps an ON object pixel, then the image pixel at the ***origin*** is set to ON, i.e. it becomes object pixel.

Arguments

Argument	Description
Point	Center point of rectangle, which specifies the tool’s work area.
Width	Rectangle width in pixels.
Height	Rectangle height in pixels.
Down threshold	Lower binarization threshold.
Up threshold	Upper binarization threshold.
Bgnd color	Background binarization color (multiple of 8).
Object color	Object binarization color (multiple of 8).
Struct. Element	Type of structuring element – 3x3 diamond, 3x3 box, 3x3 horizontal, 3x3 vertical.
Operation	Tool operation – erosion, dilation, opening or closing.

Steps	Number of operation steps (valid for all types of operations).
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Error code	Tool error code (0=OK).

Similar tools

Usually combined with

The “Find Blobs” tool.

Tips & Tricks

Call “Erosion/dilation” to get rid of “salt and pepper” noise and to receive clear and distinct objects in a binary image. Then call “Find Blobs” to get object features. Use blob binarization thresholds, which correspond to ‘Bgnd color’ and ‘Object color’ to obtain the same objects, for example:

Tool	Arguments	Comment
1. Erosion/dilation	Bgnd color = 192 Object color = 64	The tool generates result image with dark objects on light background.
2. Find Blobs	Down threshold = 60 Up threshold = 70	The blob tool detects and processes objects with color 64 (inside the range [60,70]). The range could be [64,64] as well.

Examples

Not indexed yet – sorry.

2.26. Contrast

Group

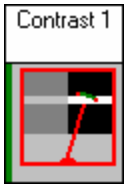
Image-processing tools

Short description

The tool calculates image contrast.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool calculates the contrast of a rectangle image area, defined by a center point, width and height arguments. The tool may work with horizontal and vertical sub-sampling, specified by the arguments “Horizontal step” and “Vertical step”. The “Mode” argument specifies tool’s operation (currently contrast only).

Results:

The tool returns 1 float result – the calculated contrast value in the range [0,100].

Algorithm

The tool first calculates `mean` (the image mean value) and `var` (the image variance value). The result contrast is calculated as:

```
contrast = SQRT(var – mean * mean)
```

Finally the tool adjusts `contrast` in the range [0,100].

Arguments

Argument	Description
Point	Center point of rectangle, which specifies the tool’s work area.
Width	Rectangle width in pixels.

Height	Rectangle height in pixels.
Horizontal step	Horizontal sub-sampling step (default 1: no sub-sampling)
Vertical step	Vertical sub-sampling step (default 1: no sub-sampling)
Mode	Mode of tool operation – currently contrast only.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Result	Calculated contrast value.

Similar tools

Light-balance.

Usually combined with

Tips & Tricks

You may find best shutter value by a tool loop, which finds best (highest) contrast for a range of shutter values with a given step. You should keep somewhere (in the point-list for example) several parameters – shutter range `[sh_min,sh_max]`, shutter step `sh_step`, current shutter value, best shutter value and best contrast value. Use the "Calculator" tool to make calculations. The suggested tool loop is:

- Select next shutter value (us) by adding `sh_step` to previous shutter value. Check for end of loop (current shutter value > `sh_max`).
- Take image with "Shutter" argument linked to current shutter value.
- Calculate contrast. If the contrast is greater than the previous maximum contrast, save new best contrast and new best shutter, equal to current shutter.

Use the found best shutter when taking image in your next processing.

Examples

Not indexed yet – sorry.

2.27. Edge Finder

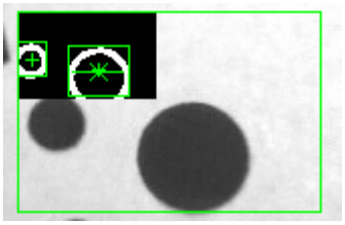
Group

Image-processing tools

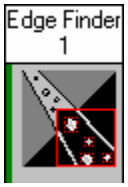
Short description

The tool finds object edges by blobs.

VIMOS kernel presentation



Editor icons



Description

General function :

The tool finds edges (contours) as blob objects. To achieve high execution speed, the tool first performs an optional pyramid operation, which decreases the image size 4 times. The next operation is a “Sobel” filter, which enhances object edges. Finally the tool searches object contours by blob analysis with binarization threshold “**Threshold**”. This tool works in a rectangular workspace, specified by the “**Point**”, “**Width**” and “**Height**” arguments.

The best tool speed is reached in the default Editor mode:

- **Pyramid** = On
- **Sobel** = Fast IMG62X
- **Show** = None
- Disable drawing of tool results.

To preserve the input image (which may be sent to PC via “Send image” tool), select "**Show**" = None. Remember that the VC Sobel option needs much lower "**Threshold**" value compared to the fast Sobel option (the VCLIB Sobel generates less distinct edges).

***ATTENTION**. The tool does not work on ADSP cameras.*

Results :

A set of results is produced for each object - center of surrounding rectangle, width, height, area, center of mass and a peripheral point. This information is stored in the point-list buffer. One point-list item is filled for each object. Each point-list item has place for 1 point and 8 floating-point values. Use the ”Get Blob from Point-list” tool to get BLOB results from the point-list.

Point-list item usage :

Item field	Object data
Point	Bounding rectangle center point
Parameter 0	Bounding rectangle width
Parameter 1	Bounding rectangle height
Parameter 2	Bounding rectangle orientation angle – not available.
Parameter 3, Parameter 4	Center of mass (x,y)
Parameter 5	Object area (pixels)
Parameter 6, Parameter 7	Peripheral point (x,y) from the object contour – rightmost point in the lowest row.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Center point of the rectangular workspace
Width	Rectangle width in pixels
Height	Rectangle height in pixels
Threshold	Brightness threshold – contour pixels, generated by the “sobel” operation, have to be brighter.
Min area	Minimum area of an object – objects with area below this value are not counted and not stored in the result point-list.
Max area	Maximum area of an object – objects with area above this value are ignored. **Note:** If this value is 0 there is no upper limit.
Ptlist start	Start index in the point-list buffer where object items will be stored.
Max results	Maximum number of objects to find (and store in the point-list).
Pyramid	Specifies preliminary pyramid operation – off or on (default).
Sobel	Type of sobel operation – fast sobel from img62x.lib or standard sobel from VCLIB.
Show	Show processed image – none, sobel image or binarized image. The shows the processed image in the upper left corner of the wirk

	rectangle.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Obj_count	Number of detected objects.

Similar tools

Find blobs.

Usually combined with

Tips & Tricks

Examples

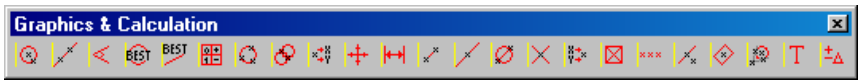
Not indexed yet – sorry.

3. Graphics & Calculations

The tools in this group are not working directly with the images. They work with variables like tool results, angles and points. Typical geometric elements like lines, circles and rectangles are supported here.

The tools may be used for calculations or for illustration purposes. Tools like ‘Marker, ‘Text box’ and ‘Rectangle’ are intended for the latter.

Editor toolbar of this group of tools:



3.1. Text Box

Group

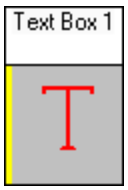
Graphics & Calculations

Short description

Able to display text and/or result values surrounded by a box in the overlay image.

VIMOS kernel presentation

Editor icons

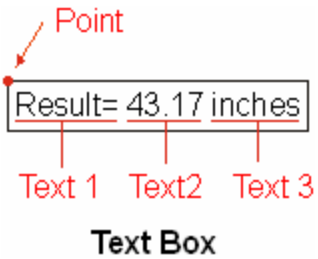


Description

General function :

Generally this is a text framed by a rectangle. It can show up to 3 strings: prefix, value and suffix. The value is multiplied by a floating-point factor (normally 1.0) and then displayed. The source of the second string (value) is a result from another tool but may be unused if only predefined text is to be displayed.

There are some predefined strings for the prefix and suffix. They may be selected directly in the VIMOS kernel (in the dialog of the tool). But since the kernel (at the moment) has no character input you have to use the VIMOS-Editor to store your own strings into the tool.



Results :

This tool has no results.

Algorithm

The tool uses the standard library functions of the camera manufacturer that supports normal English characters and numbers in two font sizes. We also have a full graphic user interface for some of the camera types (TI-processor) that allow you to display text, buttons, windows and many other elements in different colors, sizes and fonts.

Arguments

Argument	Description
Point	Defines the upper left corner of the text box
Width	Box width in pixels, 0 = size to fit
Text 1	Prefix string
Text 2	Value string (link to results from other tools, including points)
Text 3	Suffix string
Factor	Floating-point multiplier value
Text size	Size of text symbols in pixels: 1 = 8x8 pixels, 2 = 16x16 pixels
Align	Text alignment within the box: Left/Center/Right
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

This tool has no results.

Similar tools

Show string — displays a part of the string buffer. Using the tools of the ‘String tools’ group you can (pre)load strings from flash or create complex strings ‘on the fly’ and then display them.

Usually combined with

Any tool that returns at least one result - for debugging, testing and user information.

Tips & Tricks

You can link the tool position to other points. By doing this you can ‘attach’ textboxes to certain elements in the image and if they move the textboxes move with them.

The prefix and suffix strings are limited in length. So if the string that you entered in the VIMOS-Editor was to long in one of those fields please remember that you are able to specify longer strings for the string in the middle (Text2, value string).

Examples

Not indexed yet – sorry.

3.2. Marker

Group

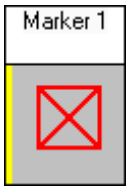
Graphics & Calculations

Short description

This tool just displays a cross (‘+’) of a given size at a given point in the overlay.

VIMOS kernel presentation

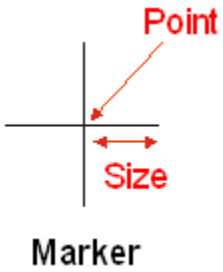
Editor icons



Description

General function :

The ‘Marker’ tool takes a point as its input and draws a cross at that place. The length of every side of the cross is defined by an additional argument.



Results :

The tool returns the point where it is actually displayed. That is done to enable a ‘read back’ of the mouse position if the tool is linked to the mouse pointer. It may of course be used for other purposes as well.

The position is returned as pixel coordinates and as ‘real world’ coordinates. For the later please read more about calibration and the ‘Select calibration set’ tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Position of the cross (marker)
Size	Length of one leg of the cross in pixels
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Point	Actual center point of the marker in pixels
Point_rw	Actual center point of the marker in 'real world' coordinates (Please read more about calibration for this matter.)

Similar tools

Coordinate system

Rectangle are other tools that have mainly display purposes. But most tools in this group may be used to display geometric elements in the overlay image.

Usually combined with

Tools that produce 'Point' results to display the location of those points. Another typical use is to link one 'Marker' tool directly to the mouse to display its virtual position.

Tips & Tricks

It's a good way to understand what happens if you use markers of different size for different elements of your program.

Examples

Not indexed yet – sorry.

3.3. Infinite Line

Group

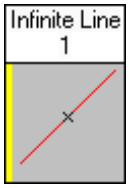
Graphics & Calculations

Short description

The tool draws a straight line at a given angle through a given point.

VIMOS kernel presentation

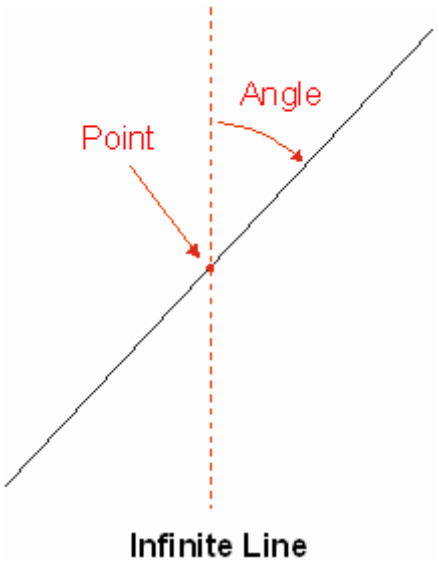
Editor icons



Description

General function :

A point and an angle define the infinite line. When the ‘Dash length’ argument is 0 a solid line is drawn, otherwise a dashed line is drawn with the specified dash length in pixels.



Results :

The Result is the angle argument. This is done to get access to it if it was linked directly to the mouse.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Point that has to be crossed by the line
Angle	Angle, defining the orientation of the line
Dash length	Line style: 0 = solid line, non-zero = dashed line
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Angle	Angle of the line (same as input if not steered or linked)

Similar tools

Finite line	draws a line between two points and returns the angle
2-point infinite line	draws an infinite line through two points and returns the angle

Usually combined with

This tool is mainly used for drawing – to visualize angular information that was acquired by other tools or is generated with this tool by linking the angle information to the mouse.

Tips & Tricks

There's nothing special to say about this tool.

Examples

Not indexed yet – sorry.

3.4. 2-Point Infinite Line

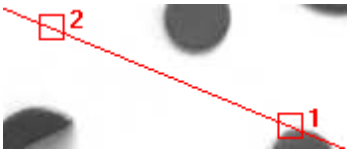
Group

Graphics & Calculations

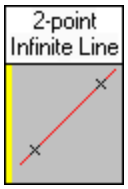
Short description

This tool draws a straight line through two given points and returns the angle as a result. The drawing is done in the overlay picture.

VIMOS kernel presentation



Editor icons



Description

General function :

This tool draws a straight line through two given points in the overlay picture. When the ‘Dash length’ argument is 0 a solid line is drawn, otherwise a dashed line is drawn with the specified dash length in pixels.

Results :

The tool returns the angle between the line and a vertical line. A ‘real world’ version of that angle is returned also. If a ‘Select calibration set’ tool has loaded a calibration set, that angle is calculated by using the X/Y relation stored in that calibration set. For instance if for every pixel in X direction the line moves one pixel in Y direction we have an angle of 45 degrees in the ‘picture coordinate system’. If no calibration set was activated the same result is returned for the ‘real world coordinate system’. But if there is an activated calibration set that tells that i.e. one millimeter in X equals to 100 pixels while one millimeter in Y equals to only 50 pixels, then it’s clear, that the pixels are not square but in real world are twice as high then wide. And so the angle of a line that rises one pixel for every pixel in horizontal direction will not be 45 degrees in real world. Please read more about calibration and the ‘Select calibration set’ tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point 1	First point that has to be crossed by the line
Point 2	Second point that has to be crossed by the line
Dash length	Line style: 0 = solid line, non-zero = dashed line
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Angle	Angle of the line
Angle_rw	Angle of the line in 'real world' coordinates (Please read more about calibration for this matter.)

Similar tools

Finite line	draws a line between two points and returns the angle
Infinite line	draws an infinite line through a point at a given angle

Usually combined with

Line cross	for finding crossing points between infinite lines
Line across circle	for finding intersections between a circle and an infinite line
Angle	for getting the angle between two infinite lines
Distance	for getting the distance between an infinite line and a point

Tips & Tricks

There's nothing special to say about this tool.

Examples

Not indexed yet – sorry.

3.5. Finite Line

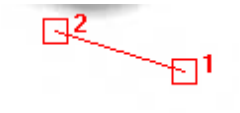
Group

Graphics & Calculations

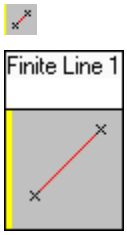
Short description

This tool draws a straight line between two given points in the overlay picture.

VIMOS kernel presentation



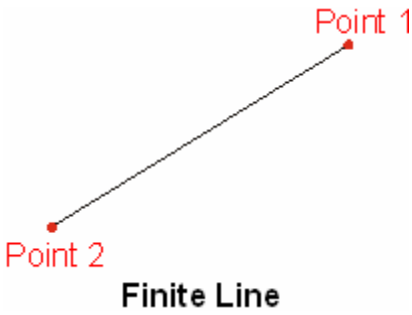
Editor icons



Description

General function :

This tool draws a straight line between two given points in the overlay picture. When the ‘Dash length’ argument is 0 a solid line is drawn, otherwise a dashed line is drawn with the specified dash length in pixels.



Results :

This tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point 1	Starting point of the line
Point 2	Ending point of the line
Dash length	Line style: 0 = solid line, non-zero = dashed line
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

This tool has no results.

Similar tools

Infinite line	draws an infinite line through a point at a given angle
2-point infinite line	draws an infinite line through two points and returns the angle

Usually combined with

This tool is mainly used for drawing.

Tips & Tricks

There's nothing special to say about this tool.

Examples

Not indexed yet – sorry.

3.6. Midpoint

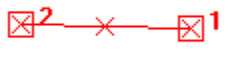
Group

Graphics & Calculations

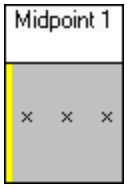
Short description

The tool returns a point, which is located at a specified position between two given points.

VIMOS kernel presentation



Editor icons



Description

General function :

Assume that the distance between two given points is equal to ‘1’ (or 100%). Now this tool allows you to generate a new point at any place between the two points if you generate a value between ‘0’ and ‘1’ or beyond the second point if you generate a value above ‘1’. To generate a value you can specify the numerator and denominator of a fraction (please see examples below). A value of ‘0.5’ will of course produce a point exactly between the two given points.

For the more mathematically oriented ☺ :

This tool produces a point that separates a given line segment at a ratio given by numerator / denominator.

So if A - start point, B - end point, n - numerator, d - denominator, then AP = (n/d) * AB, where P is the result point. Here AP and AB are vectors.

Examples : If n = 1 and d = 2, then P will be right in the middle between A and B. The same will happen with n = 50, d = 100 or n = 13, d = 26 and so on. If n = 0, then P = A. If n = d, then P = B.

If d = 0, the tool will return error code 9001 because dividing by 0 is ‘forbidden’.

Results :

The tool returns the point as described above. The position is returned as pixel coordinates and as ‘real world’ coordinates. For the later please read more about calibration and the ‘Select calibration set’ tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point 1	Start point (A)
Point 2	End point (B)
Numerator	Numerator (n)
Denominator	Denominator (d)
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Point	Generated point in pixel coordinates
Point_rw	Generated point in ‘real world’ coordinates (Please read more about calibration for this matter.)

Similar tools

Line cross	for finding crossing points between infinite lines
Circle cross	for finding crossing points between circles
Line across circle	for finding intersections between a circle and an infinite line
Point projection	for generating the point on a given line that is closest to a given point
Tangent	for generating specific points defined by a given circle and a given point
Make point	for generating a point out of separate X and Y coordinates

Usually combined with

All tools that use points as inputs.

Tips & Tricks

The higher values you use for the denominator, the finer you can specify the position.

Examples

Not indexed yet – sorry.

3.7. Best Line

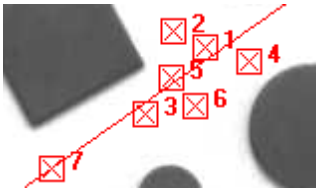
Group

Graphics & Calculations

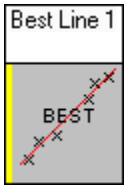
Short description

The tool generates a line through a group of points in such a way that is it as close as possible to all the points.

VIMOS kernel presentation



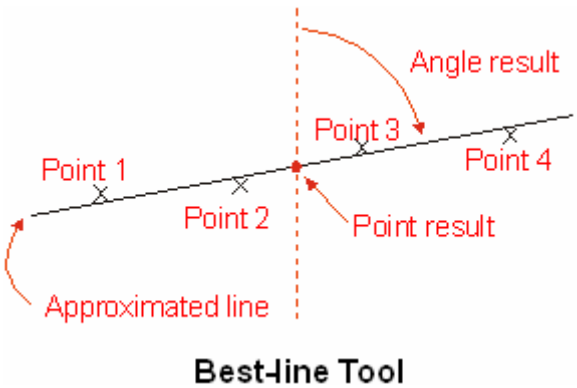
Editor icons



Description

General function :

The tool makes an approximation of several points with one infinite line. Up to ten input points can be used in the approximation. Please have a look at the ‘Point-list best line’ tool if you need more points. The argument ‘# of points’ specifies the number of points to use. The valid values are from 2 to 10. For example, if this argument is set to 5, points 1 to 5 will be used and the others will be ignored.



Results :

The tool returns results that describe an infinite line:

- a point on the generated line
- angle of the line in the image coordinate system (pixels)
- angle of the line in the 'real world' coordinate system

For the later please read more about calibration and the 'Select calibration set' tool.

Algorithm

The line is placed in such a way that the sum of all the errors is at a minimum. Here as error we use the distance between any of the points and the line.

Arguments

Argument	Description
Point 1 – Point 10	Points that define the line
# of points	Number of points to use
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Point	Point on the generated infinite line
Angle	Angle of the generated infinite line
Angle_rw	Angle of the generated infinite line in 'real world' coordinates (Please read more about calibration for this matter.)

Similar tools

Point-list best line for the same purpose but using points in the point-list buffer

Hough line

Point-list Hough line

Usually combined with

All tools that handle lines (point and angle) or one of the components (point, angle).

Tips & Tricks

To see how the tool behaves just make a small program where you place a 'Best line' tool with two or three points fixed on the screen and link one additional point to the mouse. When you start this you'll be able to move that point and see what happens to the line.

Examples

Not indexed yet – sorry.

3.8. Point Projection

Group

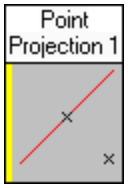
Graphics & Calculations

Short description

This tool locates the point on a given infinite line that is closest to a given point. In other words it finds the crossing point between a given line and the orthogonal line that would go through a given point.

VIMOS kernel presentation

Editor icons



Description

General function :

Calculates the orthogonal projection of a point on a line. The line is defined by a point and an angle. Please read the ‘Short description’ above for another explanation.

Results :

The tool returns the point as described above. The position is returned as pixel coordinates and as ‘real world’ coordinates. For the later please read more about calibration and the ‘Select calibration set’ tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point 1	Point to be projected
Point 2	Point from the line
Angle	Angle of the line

Dash length	Line style: 0 = solid lines, non-zero = dashed lines
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Point	Generated point in pixel coordinates
Point_rw	Generated point in 'real world' coordinates (Please read more about calibration for this matter.)

Similar tools

Line cross	for finding crossing points between infinite lines
Circle cross	for finding crossing points between circles
Line across circle	for finding intersections between a circle and an infinite line
Midpoint	for generating a point between two given points
Tangent	for generating specific points defined by a given circle and a given point
Make point	for generating a point out of separate X and Y coordinates

Usually combined with

All tools that use points as inputs.

Tips & Tricks

If you need to generate an infinite line orthogonal to a given line and through a given point you can use this tool to generate the crossing point on the line and then use a '2-point infinite line' tool to generate the wanted line itself. Please just make sure the given point is not too close or even on the given line. If that is the case you need to use a 'Calculator' tool on the angle.

Examples

Not indexed yet – sorry.

3.9. Distance

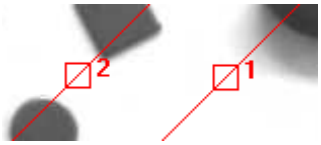
Group

Graphics & Calculations

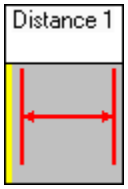
Short description

The tool is used to measure distances between points or parallel lines. Different kinds of distances are returned – measured in the image coordinate system and in ‘real world’ coordinates.

VIMOS kernel presentation



Editor icons

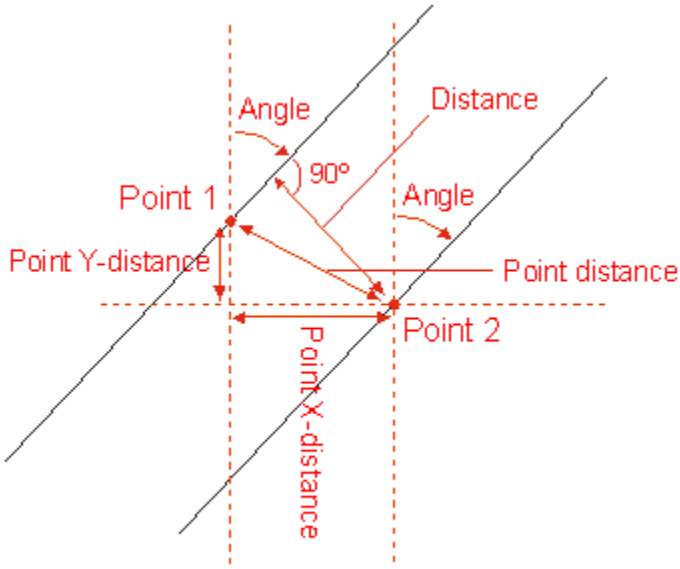


Description

General function :

The distance-tool uses as its input arguments two points and one angle. Two infinite lines are drawn through the two points at the given angle.

The tool delivers 8 results, which represent various types of distances between the two lines in pixels and ‘real world’ measuring units. For the later please read more about calibration and the ‘Select calibration set’ tool.



Distance Tool

Results :

- Distance between the two lines in pixels
- X distance between the two points |Point1.x – Point2.x| in pixels
- Y distance between the two points |Point1.y – Point2.y| in pixels
- Distance between the two points in pixels
- Distance between the two lines in real world units
- X distance between the two points |Point1.x – Point2.x| in real world units
- Y distance between the two points |Point1.y – Point2.y| in real world units
- Distance between the two points in real world units

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point 1	Point, defining the first line
Point 2	Point, defining the second line
Angle	Angle of both lines
Dash length	Line style: 0 = solid lines, non-zero = dashed lines
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Dist. between lines	Distance between the two lines in pixels
X dist. between pts.	X distance between the two points |Point1.x – Point2.x| in pixels
Y dist. between pts.	Y distance between the two points |Point1.y – Point2.y| in pixels
Dist. between pts.	Distance between the two points in pixels
Dist. between lines (rw)	Distance between the two lines in 'real world' units (Please read more about calibration for this matter.)
X dist. between pts. (rw)	X distance between the two points |Point1.x – Point2.x| in 'real world' units (Please read more about calibration for this matter.)
Y dist. between pts. (rw)	Y distance between the two points |Point1.y – Point2.y| in 'real world' units (Please read more about calibration for this matter.)
Dist. between pts. (rw)	Distance between the two points in 'real world' units (Please read more about calibration for this matter.)

Similar tools

Point-list distance	measures the distances between many points in the point-list and a given point or line

Usually combined with

Textbox	to display the distance
Send result	to transfer the distance via RS232
Tolerance	to check, if the distance is within a given range

Tips & Tricks

The tool draws its own points and lines in the overlay picture. On the other hand these elements may already have be drawn by preceding tools. Since drawing takes time you may consider hiding either this tool or the other tools that draw the same things. Before hiding something please just make sure it really works under all conditions.

Examples

Not indexed yet – sorry.

3.10. Line-cross

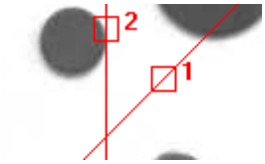
Group

Graphics & Calculations

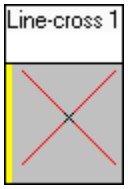
Short description

This tool just generates the crossing point between two infinite lines.

VIMOS kernel presentation



Editor icons



Description

General function :

Calculates the crossing point of two infinite lines, each one defined by a point and an angle.

Results :

The tool returns the point as described above. The position is returned as pixel coordinates and as ‘real world’ coordinates. For the later please read more about calibration and the ‘Select calibration set’ tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point 1	Point on the first line
Point 2	Point on the second line
Angle 1	Angle of the first line

Angle 2	Angle of the second line
Dash length	Line style: 0 = solid lines, non-zero = dashed lines
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Point	Generated point in pixel coordinates
Point_rw	Generated point in 'real world' coordinates (Please read more about calibration for this matter.)

Similar tools

Circle cross	for finding crossing points between circles
Line across circle	for finding intersections between a circle and an infinite line
Midpoint	for generating a point between two given points
Point projection	for generating the point on a given line that is closest to a given point
Tangent	for generating specific points defined by a given circle and a given point
Make point	for generating a point out of separate X and Y coordinates

Usually combined with

All tools that use points as inputs.

Tips & Tricks

The tool draws its own points and lines in the overlay picture. On the other hand these elements may already have be drawn by preceding tools. Since drawing takes time you may consider hiding either this tool or the other tools that draw the same things. Before hiding something please just make sure it really works under all conditions.

Examples

Not indexed yet – sorry.

3.11. Angle

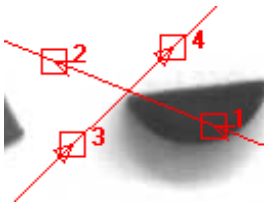
Group

Graphics & Calculations

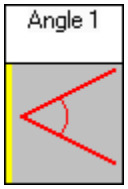
Short description

Calculates the angle between two infinite lines.

VIMOS kernel presentation



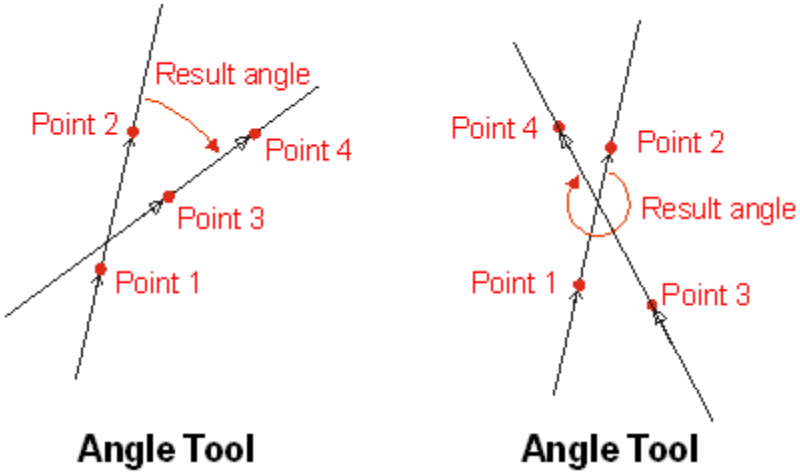
Editor icons



Description

General function :

This tool uses four points as input arguments, which define two lines. Point 1 and Point 2 define the first line and Point 3 and Point 4 define the second line.



Results :

The tool result is the angle between the two lines. The tool returns two angles: an angle, measured on the picture in pixels and an angle, measured in 'real world' coordinates. For the later please read more about calibration and the 'Select calibration set' tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point 1	First point on the first line
Point 2	Second point on the first line
Point 3	First point on the second line
Point 4	Second point on the second line
Dash length	Line style: 0 = solid lines, non-zero = dashed lines
Results	Two angles in pixel and real world units
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Angle	Measured angle
Angle_rw	Measured angle in the 'real world' coordinate system (Please read more about calibration for this matter.)

Similar tools

Point-list angle	virtually creates lines between a given point and many points in the point-list buffer and then for every such line measures returns the angle between it and a vertical line

Usually combined with

Textbox	to display the angle
Send result	to transfer the angle via RS232
Tolerance	to check, if the angle is within a given range

Tips & Tricks

The tool draws its own points and lines in the overlay picture. On the other hand these elements may already have be drawn by preceding tools. Since drawing takes time you may consider hiding either this tool or the other tools that draw the same things. Before hiding something please just make sure it really works under all conditions.

Examples

Not indexed yet – sorry.

3.12. Rectangle

Group

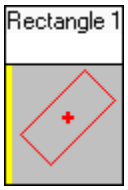
Graphics & Calculations

Short description

Draws a rectangle in the overlay picture.

VIMOS kernel presentation

Editor icons

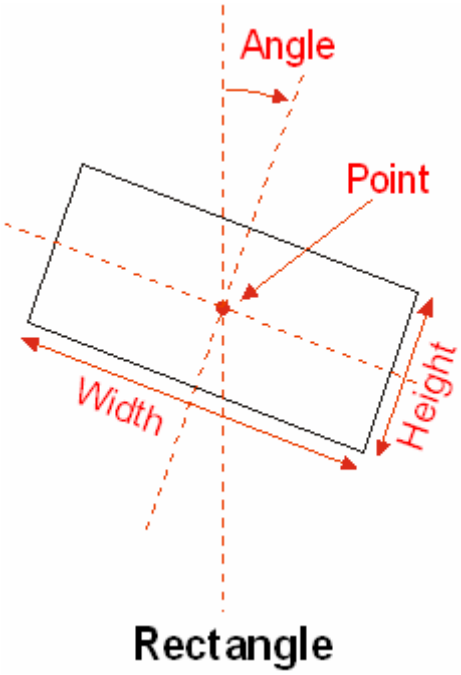


Description

General function :

This is one of the first tools and may be superfluous already because it’s possible to do the same thing by using other tools. However it would need more then one other tool and this tool is preserved for compatibility with older versions. It was made for half-automatic manual inspection systems.

The tool is intended for the drawing of rectangles in the overlay picture. The rectangle is defined by a center point, an angle and the size of its sides. All these arguments are link-able.



Results :

The tool returns rectangle area and rectangle circumference, both in 'real world' units. Please read more about calibration and the 'Select calibration set' tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Rectangle center point
Angle	Angle, specifying the rectangles rotation
Width	Rectangle width in pixels
Height	Rectangle height in pixels
Dash length	Rectangle line style: 0 = solid line, non-zero = dashed line
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Area	Area of the rectangle in the 'real world' coordinate system

	(Please read more about calibration for this matter.)
Circumference	Circumference of the rectangle in the ‘real world’ coordinate system (Please read more about calibration for this matter.)

Similar tools

Marker	draws a marker at a given point
Infinite line	draws an infinite line given by a point and an angle
2-point infinite line	generates an infinite line given by two points
Finite line	draws a line between two points
2-point circle	generates a circle given by its center and one peripheral point
Circle	generates a circle given by three of its points
Coordinate system	draws a 2D coordinate system
Textbox	draws text

Usually combined with

The tool in its present form is usable for static drawings on the screen i.e. to divide the screen into different areas for illustration purposes.

It could be used also dynamically to frame located areas of interest in the image. In that case the tool should be used together with tools that produce points or angles.

Tips & Tricks

There’s nothing special to say about this tool.

Examples

Not indexed yet – sorry.

3.13. Coordinate System

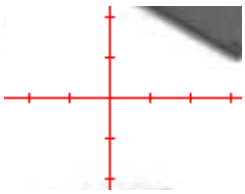
Group

Graphics & Calculations

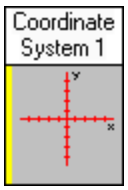
Short description

Draws a coordinate system in the overlay picture. Step width and numeration are configurable.

VIMOS kernel presentation



Editor icons



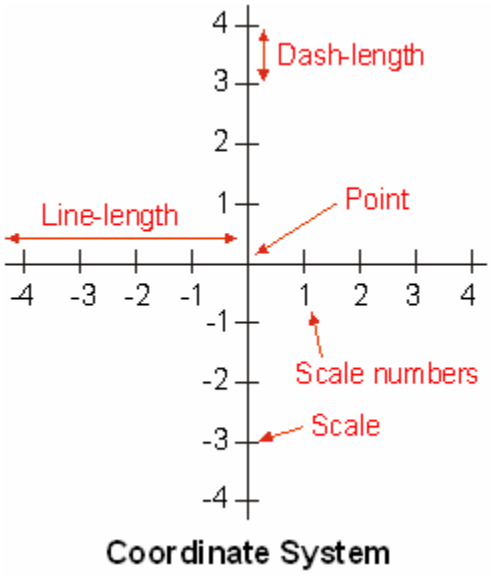
Description

General function :

This tool was made for half-automatic manual inspection systems. It draws a coordinate system in the overlay picture. A center point and several style arguments define the coordinate system.

The ‘Line-style’ argument specifies the drawing style of the axes :

- Solid – solid lines
- Dashed – dashed lines
- Scaled – solid lines with scale on them
- NumScale – like Scaled plus numbers



Results :

The tool returns the center point. So if the tool is linked to the mouse the mouse point becomes available as an input to other tools. The same is now possible with the 'Marker' tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Defines the origin point (0,0) of the coordinate system
Line style	Style of the coordinate axes
Width	Rectangle width in pixels
Dash length	Line style: 0 = solid lines, non-zero = dashed lines (distance between cross-lines for scale)
Line length	The length of each axis (0 = axes are infinite)
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Point	Center point of the coordinate system
Point_rw	Center point in the 'real world' coordinate system

	(Please read more about calibration for this matter.)

Similar tools

Marker	draws a marker at a given point
Infinite line	draws an infinite line given by a point and an angle
2-point infinite line	generates an infinite line given by two points
Finite line	draws a line between two points
Rectangle	draws a rectangle
2-point circle	generates a circle given by its center and one peripheral point
Circle	generates a circle given by three of its points
Textbox	draws text

Usually combined with

The tool is usable for illustration purposes. It would be used together with tools that produce points or angles.

Tips & Tricks

There's nothing special to say about this tool.

Examples

Not indexed yet – sorry.

3.14. Circle

Group

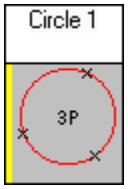
Graphics & Calculations

Short description

The ‘Circle’ tool generates a circle that crosses all the three given points.

VIMOS kernel presentation

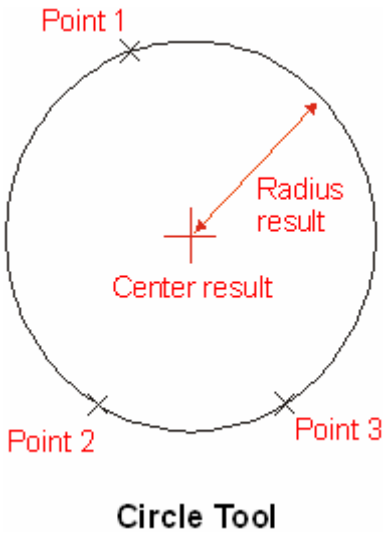
Editor icons



Description

General function :

A circle is fully defined by three points on its curve. Of course the special case of three points exactly on one straight line does NOT define a circle and should be avoided. This tool generates a circle out of three given points that are assumed to be located on the circle itself.



Results :

The tool returns the parameters describing the circle (center point and radius) and its area in ‘real world’ units. For the later please read more about calibration and the ‘Select calibration set’ tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point 1 – Point 3	Three points that define the circle
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Point	Center point of the circle
Radius	Radius of the circle in pixels
Radius_rw	Radius of the circle in the ‘real world’ coordinate system (Please read more about calibration for this matter.)
Area_rw	Area of the circle in the ‘real world’ coordinate system (Please read more about calibration for this matter.)

Similar tools

2-point circle	generates a circle given by its center and one peripheral point
Best circle	generates a circle out of up to 10 points on its curve
Point-list best circle	generates a circle out of up to 500 points on its curve
Hough circle	
Point-list Hough circle	

Usually combined with

Edge detection	finds a point on a search line where there is a transition between bright and dark in the image

Tips & Tricks

For best precision use points that are at as big an angular distance as possible. The ideal would be to have one point at every 120 deg. around the curve of the circle. If two of the points are close together the generated circle becomes unstable against small moves of one of these points.

Make sure you avoid the special case where the three points are on one straight line. There is no circle with a ‘straight curve’ ☺ and so the tool will return an error.

Examples

Not indexed yet – sorry.

3.15. 2-Point Circle

Group

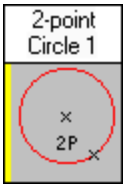
Graphics & Calculations

Short description

The tool generates a circle given by its center point and one point on the circles curve.

VIMOS kernel presentation

Editor icons



Description

General function :

A circle is fully defined by its center and an additional point on the circle's curve. This tool draws the so defined circle in the overlay picture.

Results :

The tool returns the radius of the circle in pixels and in the 'real world' coordinate system. For the later please read more about calibration and the 'Select calibration set' tool.

Note:

One may expect that in case that a 'Calibration set' is loaded the circle is drawn as an ellipse (at least if the X calibration is different from the Y calibration). VIMOS tools however always ignore the calibration right to the point where a result is produced.

The main reason is execution speed and a tradeoff between implementation time and usefulness. We know about this limitation and will address it as soon as possible.

So the 'real world' radius that is returned as a result is the 'real world' distance between the two given points. The circle that is drawn in the overlay has a constant radius in pixel coordinates, but would be an ellipse in the 'real world' if the camera has 'non-square' pixels (X and Y calibration is different).

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Center	Circle center point
Point	A point on the circles curve
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Radius	Radius of the circle
Radius_rw	Radius of the circle in the 'real world' coordinate system (Please read more about calibration for this matter.)

Similar tools

Circle	generates a circle given by three peripheral points
Best circle	generates a circle out of up to 10 points on its curve
Point-list best circle	generates a circle out of up to 500 points on its curve
Hough circle	
Point-list Hough circle	

Usually combined with

Find blobs	could be used to find the center with good precision
Edge detection	finds a point on a search line where there is a transition between bright and dark in the image

Tips & Tricks

There's nothing special to say about this tool.

Examples

Not indexed yet – sorry.

3.16. Best Circle

Group

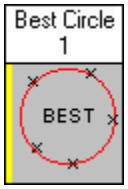
Graphics & Calculations

Short description

The tool generates a circle through a group of points in such a way that is it as close as possible to all the points.

VIMOS kernel presentation

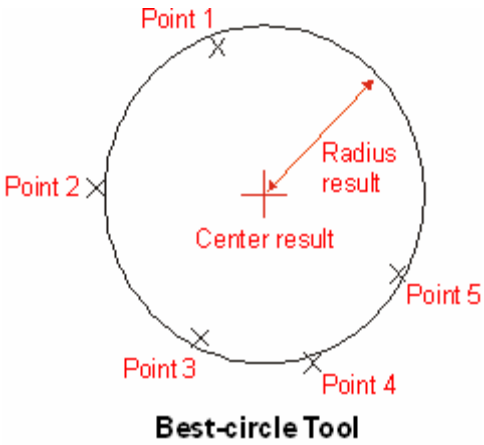
Editor icons



Description

General function :

The tool makes an approximation of several points with one circle. Up to ten input points can be used in the approximation. Please have a look at the ‘Point-list best circle’ tool if you need more points. The argument ‘# of points’ specifies the number of points to use. The valid values are from 3 to 10. For example, if this argument is set to 5, points 1 to 5 will be used and the others will be ignored.



Best-circle Tool

Results :

The tool returns results that describe a circle :

The center point of circle

Radius in the image coordinate system (pixels)

Radius in the ‘real world’ coordinate system

Circle area in the ‘real world’ coordinate system

For the ‘real world’ coordinates please read more about calibration and the ‘Select calibration set’ tool.

Algorithm

The line is placed in such a way that the sum of all the errors is at a minimum. Here as error we use the distance between any of the points and the circle.

Arguments

Argument	Description
Point 1 – Point 10	Points that define the circle
# of points	Number of points to use
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Point	Center of the generated circle
Radius	Radius of the generated circle
Radius_rw	Radius of the generated circle in ‘real world’ coordinates (Please read more about calibration for this matter.)
Area	Area of the generated circle in ‘real world’ coordinates (Please read more about calibration for this matter.)

Similar tools

Point-list best circle for the same purpose but using points in the point-list buffer

Hough circle

Point-list Hough circle

Usually combined with

All tools that handle circles (center point and radius) or one of the components (points, results).

Line across circle	to find the intersections between an infinite line and a circle
Circle cross	to find the intersections between two circles
Tangent	to find various specific points defined by a circle and a point
Tolerance	to check if the radius is within a specified range
Textbox	to display the radius

Tips & Tricks

To see how the tool behaves just make a small program where you place a ‘Best circle’ tool with two or three points fixed on the screen and link one additional point to the mouse. When you start this you’ll be able to move that point and see what happens to the circle.

Examples

Not indexed yet – sorry.

3.17. Circle-cross

Group

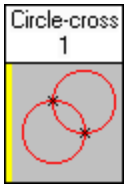
Graphics & Calculations

Short description

The tool is used to find the crossing points of two circles

VIMOS kernel presentation

Editor icons



Description

General function :

This tool returns the intersection points of two circles. Each circle is defined by its center point and radius. If the circles coincide or do not intersect, error code 9001 is returned.

Results :

Depending on size and position of the circles there are four different possibilities :

- No intersection : error code 9001 is returned
- The circles touch one another in exactly one point : two equal points are returned
- The circles intersect in two points : these points are returned
- The circles are the same – unlimited number of intersections : error code 9001 is returned

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Center_1	Center point of first circle
Center_2	Center point of second circle
Radius_1	Radius of first circle
Radius_2	Radius of second circle
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Intersection_1	Intersection point 1
Intersection_2	Intersection point 2
Intersection_1_rw	Intersection point 1 in the 'real world' coordinate system (Please read more about calibration for this matter.)
Intersection_2_rw	Intersection point 2 in the 'real world' coordinate system (Please read more about calibration for this matter.)

Similar tools

Line cross	for finding the crossing point between two infinite lines
Line across circle	for finding intersections between a circle and an infinite line
Midpoint	for generating a point between two given points
Point projection	for generating the point on a given line that is closest to a given point
Tangent	for generating specific points defined by a given circle and a given point
Make point	for generating a point out of separate X and Y coordinates

Usually combined with

All tools that use points as inputs.

Tips & Tricks

The tool draws its own points and circles in the overlay picture. On the other hand these elements may already have be drawn by preceding tools. Since drawing takes time you may consider hiding either this tool or the other tools that draw the same things. Before hiding something please just make sure it really works under all conditions.

Examples

Not indexed yet – sorry.

3.18. Tangent

Group

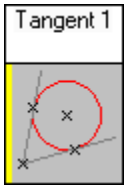
Graphics & Calculations

Short description

The tool uses a circle and point. It returns the points on the circle where the tangential lines that go through the additional point would touch the circle. The tool also returns the points of the circle that are closest to and most distant from the additional point.

VIMOS kernel presentation

Editor icons



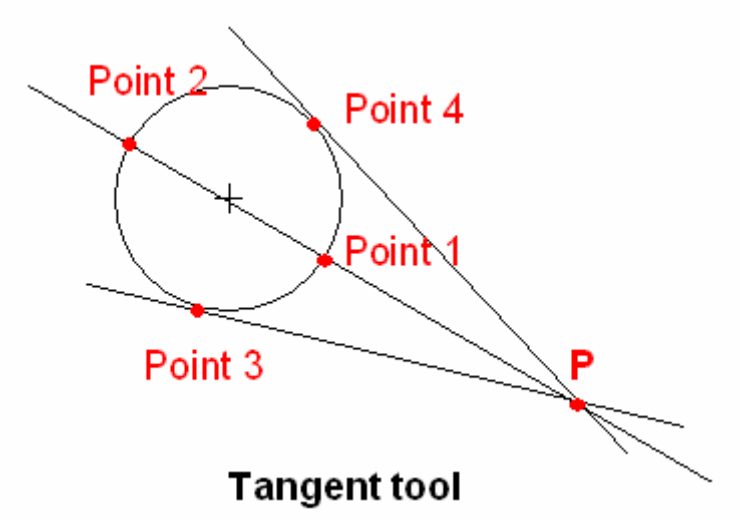
Description

General function :

This tool takes a circle (center, radius) and a point P as arguments. It produces 4 points: closest and furthest points to P, two tangent points. All 4 points lie on the circle. A tangent point is located where a tangent line through P touches the circle.

If P is inside the circle, third and fourth results contain error code 9001.

If P coincides with the circle center point, all 4 results contain error code 9001.



Results :

Please read the previous sentences.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Outside point P
Center	Center point of the circle
Radius	Radius of the circle
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Point_1	Closest circle point to P
Point_2	Furthest circle point from P
Point_3	First tangent point
Point_4	Second tangent point (could coincide with the first one)
Point_1_rw	Closest circle point to P in 'real world' coordinates (Please read more about calibration for this matter.)
Point_2_rw	Furthest circle point from P in 'real world' coordinates

	(Please read more about calibration for this matter.)
Point_3_rw	First tangent point in ‘real world’ coordinates (Please read more about calibration for this matter.)
Point_4_rw	Second tangent point in ‘real world’ coordinates (Please read more about calibration for this matter.)

Similar tools

Line cross	for finding the crossing point between two infinite lines
Circle cross	for finding crossing points between circles
Line across circle	for finding intersections between a circle and an infinite line
Midpoint	for generating a point between two given points
Point projection	for generating the point on a given line that is closest to a given point
Make point	for generating a point out of separate X and Y coordinates

Usually combined with

All tools that use points as inputs.

Tips & Tricks

The tool draws its own circle and lines in the overlay picture. On the other hand these elements may already have be drawn by preceding tools. Since drawing takes time you may consider hiding either this tool or the other tools that draw the same things. Before hiding something please just make sure it really works under all conditions.

Examples

Not indexed yet – sorry.

3.19. Line Across Circle

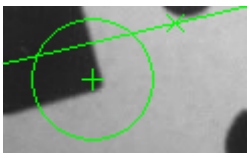
Group

Graphics & Calculations

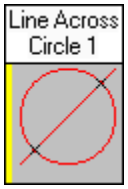
Short description

This tool returns the crossing points between an infinite line and a circle.

VIMOS kernel presentation *



Editor icons



Description

General function :

This tool calculates the intersection points of a line and a circle.

Results :

Depending on the situation there are three different possibilities :

No intersection : error code 9001 is returned

The line touches the circle in exactly one point : two equal points are returned

The line crosses the circle in two points : these points are returned

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Center	Center point of the circle
Point	A point from the line

Radius	Radius of the circle
Angle	Angle of the line
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Intersection_1	Intersection point 1
Intersection_2	Intersection point 2
Intersection_1_rw	Intersection point 1 in the 'real world' coordinate system (Please read more about calibration for this matter.)
Intersection_2_rw	Intersection point 2 in the 'real world' coordinate system (Please read more about calibration for this matter.)

Similar tools

Line cross	for finding the crossing point between two infinite lines
Circle cross	for finding intersections between two circles
Midpoint	for generating a point between two given points
Point projection	for generating the point on a given line that is closest to a given point
Tangent	for generating specific points defined by a given circle and a given point
Make point	for generating a point out of separate X and Y coordinates

Usually combined with

All tools that use points as inputs.

Tips & Tricks

The tool draws its own line and circle in the overlay picture. On the other hand these elements may already have be drawn by preceding tools. Since drawing takes time you may consider hiding either this tool or the other tools that draw the same things. Before hiding something please just make sure it really works under all conditions.

Examples

Not indexed yet – sorry.

3.20. Tolerance

Group

Graphics & Calculations

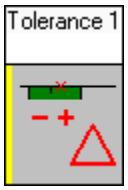
Short description

The tolerance tool is used to see if a given value is within the expected range or not.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool checks if the argument 'Value' is in the specified range [Expected – Negative tolerance, Expected + Positive tolerance]. The source of the 'Value' argument is normally another tool.

Results :

The tool returns two float results. The first result **"Comparison result"** receives one of these three values :

- -1 : BAD (too small) : Value < (Expected – Neg. tolerance)
- 0 : GOOD : (Expected – Neg. tolerance) <= Value <= (Expected + Pos. tolerance)
- 1 : BAD (too big) : Value > (Expected + Pos. tolerance)

The second result **"Valid value"** receives one of the following values:

- Min = Expected – Neg. tolerance if "Comparison result" = -1
- Value if "Comparison result" = 0
- Max = Expected + Pos. tolerance if "Comparison result" = 1

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Value	An argument, which is checked to be in the specified range. The argument should be linked to a tool result.
Point axis	Selector for X-axis or Y-axis when the **Value** argument is linked to a point-type result (ignored for non-point results)
Expected	Expected value of the **Value** argument
Pos. tolerance	Allowed positive tolerance
Neg. tolerance	Allowed negative tolerance

Results

Result	Description
Comparison result	Comparison result: -1=below, 0=OK, 1=above
Valid value	Valid result value, which is always in the allowed range.

Similar tools

There are no similar tools but you can do flexible comparisons by using IF-constructions.

Usually combined with

All tools that produce results, especially results in the ‘real world’ coordinate system.

Tips & Tricks

There is nothing special to say about this tool.

Examples

Not indexed yet – sorry.

3.21. Calculator

Group

Graphics & Calculations

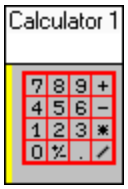
Short description

The calculator tool is able to do calculations between a tools result and either a fixed coefficient or another tools result.

VIMOS kernel presentation

This tool does not have a graphical presentation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The calculator tool makes calculations between the left-hand operand "**Operand_1**" and one of the right-hand operands "**Operand_2a**" or "**Operand_2b**". The left-hand operand should always be linked to a tool result (you can't use a constant). The right-hand operand is selected by the "**Mode**" argument - another tool result from "**Operand_2a**" or a constant value from "**Operand_2b**". Single-operand calculations like `sin` and `sqrt` use "**Operand_1**".

The tool performs the following calculations, selected by the "**Operator**" argument:

- `'+', '-', '*', '/', mod`
- `sin, cos, tan, cotan, arcsin, arccos, arctan, arccotan`
- `sqrt`
- logical operations `AND, OR, XOR`.

Results :

The tool just returns the result of the calculation.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Operand_1	Left-hand operand – should always be linked to a tool result
Operator	Calculation operator (+, -, *, /, ...)
Operand_2a	Right-hand operand from another result - should be linked to a tool result
Operand_2b	Right-hand operand from a constant value
Mode	Select right-hand operand – Operand_2a or Operand_2b

Results

Result	Description
Result	Result of the calculation

Similar tools

There are no similar tools.

Usually combined with

All tools that produce results.

Tips & Tricks

The IF-tools do not directly allow comparing two result values, but only a result against a fixed value. To overcome this you can use a ‘Calculator’ tool and subtract one of the result values from the other. The result will be less, equal or bigger then zero depending on the relation between the two values. So you can use the result of the ‘Calculator’ tool in the IF-tool and compare it there i.e. against zero.

Examples

Not indexed yet – sorry.

3.22. Convert Point

Group

Graphics & Calculations

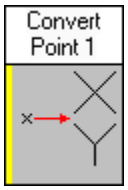
Short description

Extracts the X and Y coordinates out of a point argument.

VIMOS kernel presentation

This tool does not have a graphical presentation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

This tool takes a point as an argument and outputs its X and Y coordinates as two separate results.

Results :

The tool just returns the two coordinates. They may be generated in pixels or as 'real world' coordinates. For the later please read more about calibration and the 'Select calibration set' tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Input point
Coordinates	Coordinate transformation to apply : No change / Pixels to real-world / 'Real world' to pixels

Results

Result	Description
X	X coordinate

Y	Y coordinate

Similar tools

Make point	reverse operation – creates a point out of given coordinates

Usually combined with

Calculator	together with ‘Make point’ allows to ‘displace’ a point with a given offset
Tolerance	could be used to see if the position of the point is OK

Tips & Tricks

If you need store a really huge number of points so that the remaining place in the point-list buffer would not be able to hold them you could try to use ‘Convert point’ in a cycle and store the generated coordinates in the parameter fields of the point-list buffer. Since there are eight such fields per entry you will be able to store four additional points per entry.

That operation would be very helpful when you intend to save this point-list to flash memory because the parameter fields are always stored with the entry and so the file will be much shorter.

Note : The described operation would require to execute the tool in a cycle and that takes time. Especially the ADSP-based cameras would be quite slow because they would have to reload different parts of VIMOS during every cycle. The TI-based cameras will do better.

Examples

Not indexed yet – sorry.

3.23. Make Point

Group

Graphics & Calculations

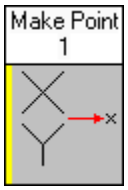
Short description

Creates a new point out of two given coordinate values (X,Y).

VIMOS kernel presentation

This tool does not have a graphical presentation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

This tool constructs a point from given X & Y coordinates. They may be given in pixels or as 'real world' coordinates. For the later please read more about calibration and the 'Select calibration set' tool.

Results :

The tool just returns the point.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
X coordinate	X coordinate of the point
Y coordinate	Y coordinate of the point
Coordinates	Coordinate transformation to apply : No change / Pixels to real-world / 'Real world' to screen

Results

Result	Description
P	Point result

Similar tools

Convert point reverse operation – returns the coordinates of a point

Usually combined with

All tools that use points.

Tips & Tricks

If you need store a really huge number of points so that the remaining place in the point-list buffer would not be able to hold them you could try to use ‘Convert point’ in a cycle and store the generated coordinates in the parameter fields of the point-list buffer. Since there are eight such fields per entry you will be able to store four additional points per entry.

That operation would be very helpful when you intend to save this point-list to flash memory because the parameter fields are always stored with the entry and so the file will be much shorter.

Note : The described operation would require to execute the tool in a cycle and that takes time. Especially the ADSP-based cameras would be quite slow because they would have to reload different parts of VIMOS during every cycle. The TI-based cameras will do better.

Examples

Not indexed yet – sorry.

4. Point-list tools

The tools from this group work with lists of points. Each point-list ***item*** (also called ***element***) is composed of several fields - 1 point and 8 floating-point numbers. The number fields in the item structure are called further item ***parameters*** and are denoted by “Param 0” to “Param 7”. Point-lists are stored in a VIMOS kernel memory buffer called the ***Point-list buffer***. In general tools from this group read input data from the point-list buffer and/or store results into the point-list buffer.

Editor toolbar of this group of tools :

4.1. Load Point-list

Group

Point-list tools.

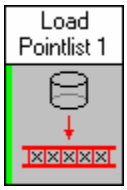
Short description

The tool loads the point-list buffer from flash file.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

This tool loads a section in the point-list buffer from a flash file. The argument "Start index" specifies start location in the point-list buffer where file data is stored. The argument "File" specifies file identifier (number) – 0 for file ***pl0.vm***, 1 for ***pl1.vm***, etc. This tool works only in **run-mode**.

Results:

The tool returns 2 float results – number of loaded point-list items (section size) and next free location in the point-list buffer.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Start position in the point-list buffer.
File	Identifier (number) of point-list file ***pl***<xxx>***.vm***, where xxx is number from 0 to 999.

Results

Result	Description
Section size	Number of point items loaded from file.
Next position	Position of next free point-list item after loaded section (“Start index” + “Section size”).

Similar tools

Save Point-list.

Usually combined with

Save Point-list.

Tips & Tricks

Examples

Not indexed yet – sorry.

4.2. Save Point-list

Group

Point-list tools.

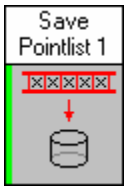
Short description

The tool saves a part of the point-list buffer to flash file.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

This tool saves a section from the point-list buffer to flash file. The argument "Start index" specifies start position of the section in the point-list buffer. The argument "File" specifies file identifier (number) – 0 for file ***pl0.vm***, 1 for ***pl1.vm***, etc. This tool works only in **run-mode**.

ATTENTION*. This tool writes a file, which may quickly consume all available flash space on the camera.*

Results:

The tool has no results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Start position in the point-list buffer.
Section size	Number of point-list items to save.
File	Identifier (number) of point-list file ***pl***<xxx>***.vm***, where xxx is

	number from 0 to 999.

Results

The tool has no results.

Similar tools

Load Point-list.

Usually combined with

Load Point-list.

Tips & Tricks

Examples

Not indexed yet – sorry.

4.3. Set Point-list Parameter

Group

Point-list tools.

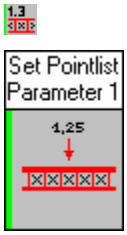
Short description

The tool sets a point-list parameter (floating-point number) in a section of point-list items.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool sets the value of a specified parameter in a set (section) of point-list items. Tool arguments specify a value to be set, a parameter, which receives the value, a section start index in the point-list buffer and a section size. A single item parameter can be set when section size is equal to of 1. The parameter value can be a constant, specified by the configuration dialog or dynamic value, received by linking the argument "Dynamic value" to a tool's result.

Results:

The tool returns one float result equal to the input parameter value.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Start position of the section in the point-list buffer (index of first section item).
Section size	Number of point-list items in the section.
Constant value	A value to be set into specified parameter when "Mode" = "Constant value".

Dynamic value	A value to be set into specified parameter when “Mode” = “Dynamic value”. If used, this argument should be linked to a tool result.
Mode	Selects which value to set - “Constant value” or “Dynamic value”.
Set value in	Selects point-list item parameter which receives the value – “Param 0” to “Param 7”.

Results

Result	Description
Parameter value	The input parameter value - “Constant value” or “Dynamic value”.

Similar tools

Get Point-list Parameter	Gets parameter from a point-list item
Set Point-list Item	Sets some or all fields in a point-list item
Get Point-list Item	Gets all fields from a point-list item

Usually combined with

Get Point-list Parameter

Tips & Tricks

Examples

Not indexed yet – sorry.

4.4. Set Point-list Item

Group

Point-list tools.

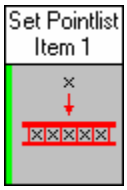
Short description

The tool sets some or all fields (point and parameters) in a point-list item.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool sets one item in the point-list buffer. You may load all item fields (the point and the 8 float parameters) or part of the fields – point only or parameters only.

.

Results:

The tool has no results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Point-list index	Item index in the point-list buffer.
Point	Point argument, which is stored into item's point when "Update" = "All" or "Point".
Param 0	Value stored into item's parameter 0 when "Update" = "All" or "Parameters".
Param 1	Value stored into item's parameter 1 when "Update" = "All" or "Parameters".
Param 2	Value stored into item's parameter 2 when "Update" = "All" or

	"Parameters".
Param 3	Value stored into item's parameter 3 when "Update" = "All" or "Parameters".
Param 4	Value stored into item's parameter 4 when "Update" = "All" or "Parameters".
Param 5	Value stored into item's parameter 5 when "Update" = "All" or "Parameters".
Param 6	Value stored into item's parameter 6 when "Update" = "All" or "Parameters".
Param 7	Value stored into item's parameter 7 when "Update" = "All" or "Parameters".
Update	Specifies which tool arguments are loaded into item fields: • **All**. Load point and parameters. • **Point**. Load point only. • **Parameters**. Load "Param 0" to "Param 7" only.

Results

The tool has no results.

Similar tools

Set Point-list Parameter	Sets parameter in a section of point-list items
Get Point-list Parameter	Gets parameter from a point-list item
Get Point-list Item	Gets all fields from a point-list item

Usually combined with

Get Point-list Item

Tips & Tricks

Examples

Not indexed yet – sorry.

4.5. Get Point-list Parameter

Group

Point-list tools.

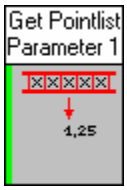
Short description

The tool gets a parameter from a point-list item.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool reads one parameter from a point-list item and returns its value. Tool arguments specify item's index in the point-list buffer and the parameter, which should be read.

Results:

The tool returns one float result - the value of the specified parameter.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Element index	Index of point-list buffer item.
Read value from	Selects parameter to be read – "Param 0" to "Param 7".

Results

Result	Description
Parameter value	Value of specified parameter.

Similar tools

Set Point-list Parameter	Sets parameter in a section of point-list items
Set Point-list Item	Sets some or all fields in a point-list item
Get Point-list Item	Gets all fields from a point-list item

Usually combined with

Tools that store results in the point-list buffer.

Tips & Tricks

Examples

Not indexed yet – sorry.

4.6. Get Point-list Item

Group

Point-list tools.

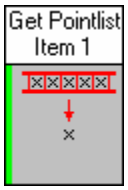
Short description

The tool returns as results all fields (1 point and 8 float numbers) of a point-list item.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool reads all fields from a point-list item – 1 point and 8 float parameters.

Results:

The tool returns values of item fields into 1 point result and 8 float results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Point-list index	Index of point-list item (item's position in the point-list buffer).

Results

Result	Description
Point	Item's point
Param 0	Item's parameter 0.
Param 1	Item's parameter 1.
Param 2	Item's parameter 2.
Param 3	Item's parameter 3.
Param 4	Item's parameter 4.
Param 5	Item's parameter 5.
Param 6	Item's parameter 6.
Param 7	Item's parameter 7.

Similar tools

Set Point-list Parameter	Sets parameter in a section of point-list items
Get Point-list Parameter	Gets parameter from a point-list item
Set Point-list Item	Sets some or all fields in a point-list item

Usually combined with

Tools that store results in the point-list buffer.

Tips & Tricks

Examples

Not indexed yet – sorry.

4.7. Get Blob from Point-list

Group

Point-list tools.

Short description

The tool gets information of one blob object from the point-list.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons

Description

General function:

This tool reads the parameters of a blob object from a point-list item. The "Find Blobs" tool stores information about each found object into one point-list item. A blob object is defined by a bounding rectangle (rectangle which encloses the object, currently not rotated), center of object mass, object area and a point, which lies on the object's contour. You may use also "Get Point-list Item" tool to extract object information, but this tool is more convenient, because it returns results with proper types.

Results:

The tool returns several point, float and angle results with object parameters (details in "Results" table below).

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Point-list index	Index of a point-list item, which contains information about one blob object.

Results

Result	Description
Center point (P)	Center point of bounding rectangle.
Rectangle width (R)	Width of bounding rectangle.
Rectangle height (R)	Height of bounding rectangle.
Angle (A)	Rotation angle of bounding rectangle (currently not implemented).
Center of mass (P)	Center of object mass (much more precise than center of bounding rectangle, calculated with sub-pixel accuracy).
Object area (R)	Area of blob object in number of pixels.
Contour point (P)	A point from the object's contour (especially useful for linking to a "Contour" tool).

Similar tools

Get Point-list Item Gets all fields from a point-list item

Usually combined with

After "Find Blobs" tool to get blob results.

Before "Contour" tool to supply initial point for contour tracing.

Tips & Tricks

Examples

Not indexed yet – sorry.

4.8. Point-list Sort

Group

Point-list tools.

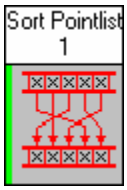
Short description

The tool sorts point-list items depending on one parameter (point coordinates or float parameters).

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool sorts in ascending or descending order the items in a section from the point-list buffer and stores the sorted items into another point-list section. Tool arguments define start indexes of source and target point-list sections, source/target section size and sorting order. The sorting operation can be done in respect to x/y coordinates of the item points or one of the 8 float parameters. The items in the source section are not modified if you don't define overlapping sections.

Results:

The tool has no results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Source_start	Beginning index of source point-list section.
Section_size	Size of source and target sections (number of items).
Target_start	Beginning index of target point-list section.
Sort_by	Specifies an item's field, according to which the point-list is sorted – point "X coord", point "Y coord" , "Param 0" to "Param 7".

Sort_order	Specifies sorting order - ascending or descending.

Results

The tool has no results.

Similar tools

Point-list Filter.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.9. Point-list Filter

Group

Point-list tools.

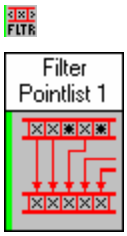
Short description

The tool filters a point-list section - it copies items into another point-list section depending on a given parameter (point coordinate or float parameter).

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool passes the items in a point-list section through a filter algorithm and stores the items that match the filter condition into another point-list section. Tool arguments define start indexes of source and target point-list sections and source/target section size. The argument "Filter by" specifies the item's parameter in respect to which the filter algorithm is applied - x/y coordinates of item points or one of the 8 float parameters. The items in the source section are not modified if you don't define overlapping sections.

The filtering is done using two numeric values - threshold 1 and threshold 2. The "Filter algorithm" argument specifies the filter condition. Point-list items, which satisfy the condition, are stored into the target section:

```
Parameter >  threshold 1
Parameter >= threshold 1
Parameter == threshold 1
Parameter != threshold 1
Parameter <= threshold 1
Parameter <  threshold 1
Parameter > threshold 1 and Parameter < threshold 2
Parameter < threshold 1 or Parameter > threshold 2
```

The last two conditions check whether selected parameter is inside or outside the range

```
[threshold 1, threshold 2]
```

Results:

The tool returns one float result - number of point-list elements that match the filter condition and are stored in the target point-list section.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Beginning index of source point-list section.
Source/target size	Size of source and target sections (number of items).
Start index	Beginning index of target point-list section.
Filter by	Specifies item's field on which the filter is applied – point "X coord", point "Y coord" and "Param 0" to "Param 7"
Constant threshold 1	Constant threshold 1 value used in filter condition.
Dynamic threshold 1	Dynamic threshold 1 value used in filter condition. If used, this argument should be linked to a tool result.
Threshold 1 mode	Type of threshold 1 value - "Constant value" or "Dynamic value".
Constant threshold 2	Constant threshold 2 value used in filter condition.
Dynamic threshold 2	Dynamic threshold 2 value used in filter condition. If used, this argument should be linked to a tool result.
Threshold 2 mode	Type of threshold 2 value - "Constant value" or "Dynamic value".
Filter algorithm	Specifies filter condition.

Results

Result	Description
Target section size	Number of point-list items that match the filter condition and are stored in the target point-list section.

Similar tools

Point-list Sort.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.10. Point-list Display

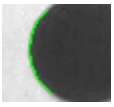
Group

Point-list tools.

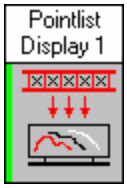
Short description

The tool provides several ways to display points from a point-list section.

VIMOS kernel presentation



Editor icons



Description

General function:

The tool displays a point-list section in the overlay or in current frame buffer (destructive mode). Tool arguments define start index of point-list section and section size. The tool draws pixels with coordinates, specified by the point fields of the point-list items. Pixel values are taken from item's float parameters or from the argument "Constant".

Results:

The tool has no results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Beginning index of point-list section in the point-list buffer.
Section size	Number of points (items) to display.
Display on:	Video-buffer where the tool draws points – into the overlay or into the frame buffer (destructive).

Gray value from	Source of pixel values – a constant value or a parameter value (“Param 0” to “Param 7”).
Constant	Constant pixel value to draw when “Gray value from” = “Constant”.
Draw_clr	Tool drawing color on success (0=default : green). **Used when drawing in the overlay.**
Err_clr	Tool drawing color on error (0=default : red). **Used when drawing in the overlay.**
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only **Used when drawing in the overlay.**

Results

The tool has no results.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.11. Generate Point-list Line

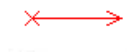
Group

Point-list tools.

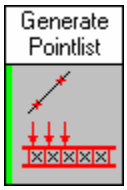
Short description

The tool generates coordinates of line points in stores results in the point-list buffer.

VIMOS kernel presentation



Editor icons



Description

General function:

The tool generates coordinates of line points and stores results in the point-list buffer. Tool arguments specify start and final line point. The line generation begins from the start point. The coordinates of the line points are calculated with sub-pixel accuracy. The argument “Number of points” specifies how many points will be generated between the start and the final point, i.e. the number of result points, stored in the point-list buffer. The minimum number of generated points is 2. The maximum number of generated points is limited by the available free space in the point-list buffer, starting from the “Start index” argument. Generated points are stored into the “point” fields of the result point-list items.

***ATTENTION**. The generated line is not a Bresenham’s line with fixed number of points, equal to max(abs(x1-x2),abs(y1-y2)), where (x1,y1) and (x2,y2) are the two end line points.*

.

The generated line is displayed on screen as arrow.

Results:

The tool has no results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start point	Beginning line point.
Final point	End line point.
Start index	Start index in point-list buffer where generated line points are stored.
Number of points	Number of generated points.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

The tool has no results.

Similar tools

Generate Point-list Circle.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.12. Generate Point-list Circle

Group

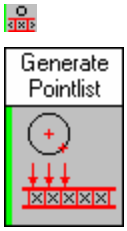
Point-list tools.

Short description

The tool generates coordinates of circle points in stores results in the point-list buffer.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool generates coordinates of circle points and stores results in the point-list buffer. Tool arguments define the circle by a center point and a circle peripheral point. The coordinates of the circle points are calculated with sub-pixel accuracy. The argument “Number of points” specifies how many circle points will be generated, i.e. the number of items, stored in the point-list buffer. The minimum number of generated points is 3. The maximum number of generated points is 6283 (also limited by the available free space in the point-list buffer, starting from the “Start index” argument). The circle generation begins from the circle point in clockwise or counterclockwise direction, defined by the argument “Sweep direction”. Generated points are stored into the “point” fields of the result point-list items.

***ATTENTION**. The generated circle is not a Bresenham’s circle with number of points, defined by the circle radius.*

.

The generated circle is displayed in the overlay. The sweep direction is shown by an arrow on the screen.

Results:

The tool has no results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Center point	Center point of the circle.
Circle point	Peripheral circle point which defines circle radius.
Sweep direction	Direction for generation of circle points starting from the peripheral circle point (clockwise or counterclockwise).
Start index	Start index in point-list buffer where generated circle points are stored.
Number of points	Number of generated points.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

The tool has no results.

Similar tools

Generate Point-list Line.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.13. Point-list Transform

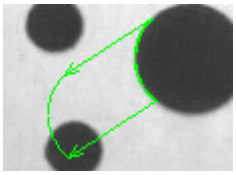
Group

Point-list tools.

Short description

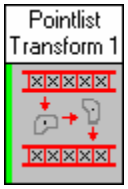
The tool translates and rotates point-list coordinates.

VIMOS kernel presentation



The tool draws the points of the source and the destination point-lists and two arrows between the first and the end source/destination points, which show the transformation. In edit mode additional arrow beginning from the reference point shows the rotation angle.

Editor icons



Description

General function:

The tool translates and rotates the source point-list section and stores result points into destination point-list section. The argument “Section size” specifies number of points in the source and destination sections. The rotation is done with center point “Reference point” and angle “Rotation angle”.

The translation can be done in two ways:

- Using the “Translation point” when “Set translation with” = “Point”. The points are translated by horizontal and vertical displacement dx and dy, necessary to move the reference point to the translation point.
- By direct horizontal and vertical displacements, specified by “Translation Dx” and “Translation Dy” (“Set translation with” = “Dx&Dy”).

The rotated and translated points are stored into the destination point-list section. Avoid overlapping of the two sections. Note that when the reference and the translation points are identical you will see the tool in VIMOS kernel as a single cross.

Results:

The tool has no results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Start index of source point-list section.
Section size	Number of source and destination points.
Start index	Start index of destination point-list section.
Reference point	Reference point for “Point” translation or rotation.
Rotation angle	Rotation angle (0 = no rotation).
Set translation with	Method of translation – by point or by (Dx, Dy)
Translation point	Translation point used when “Set translation with” = “Point”.
Translation Dx	Translation on x-axis when Dx&Dy method is used.
Translation Dy	Translation on y-axis when Dx&Dy method is used.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

The tool has no results.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.14. Point-list Read Pixels

Group

Point-list tools.

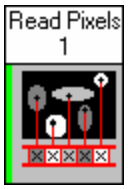
Short description

The tool reads gray-scale values at locations, specified by a point-list, and stores values into point-list parameters.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool calculates gray-scale values of a set of points, specified by a point-list section. The tool works with sub-pixels and applies bilinear interpolation to calculate pixel values in the range 0.0 to 255.0. The result value for each point is stored in a specified parameter of the point-list item (Param 0 to Param 7).

Results:

The tool returns six results – two points (the brightest and the darkest point), their indexes in the point-list buffer and their actual gray-scale values.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Start index of point-list section.
Section size	Number of section points.
Write values in	Point-list parameter which receives the pixel value – Param 0 to Param 7.

Results

Result	Description
Brightest point (P)	Point with greatest gray-scale value (brightest pixel)
Brightest index (R)	Index of brightest point in the point-list buffer.
Brightest value (R)	Gray-scale value of brightest point.
Darkest point (P)	Point with lowest gray-scale value (darkest pixel).
Darkest index (R)	Index of darkest point in the point-list buffer.
Darkest value (R)	Gray-scale value of darkest point.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.15. Point-list Median Filter

Group

Point-list tools.

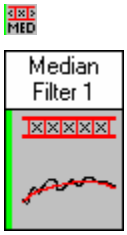
Short description

The tool applies median filter on a point-list section.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool applies a median filter on a point-list section. Input filter data is taken from the item's parameter, specified by the "Read from" argument. Filter results are stored into the item's parameter, specified by the "Write in" argument. The maximum section size is limited to 1000 items. The tool may filter non-circular or circular point-list as specified by the "Circular" argument.

Results:

The tool has no results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Start index of point-list section.
Section size	Number of section items (< 1000).
Filter size	Size of filter mask – 3, 5, 7 or 9.
Circular	Specifies non-circular or circular point-list section. Set this option to "Yes" if the section contains circular data, for example circle points generated by the "Generate Point-list Circle" tool. The tool appends the beginning items at the end of the section to perform circular

	filtering.
Read from	Specifies input parameter in the point-list items, which is filtered – Param0 to Param 7.
Write in	Specifies output parameter in the point-list items, where results are stored – Param0 to Param 7.

Results

The tool has no results.

Similar tools

Usually combined with

Apply a median filter (low pass, softening) to a sequence of values located within a given parameter of a part of the point-list buffer (point-list section). A standard way to use this would be first to fill the points in a point-list section with coordinates of pixels (for example by calling the 'Generate Point-list Circle' tool). Then use the 'Point-list Read Pixels' tool to get the actual brightness into one of the parameters of these point-list items and finally call this filter. After that the 'Point-list Edge detection' tool can be used.

Tips & Tricks

Examples

Not indexed yet – sorry.

4.16. Point-list Edge Detection

Group

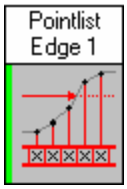
Point-list tools.

Short description

The tool detects edges in a point-list section.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool finds edges in a point-list buffer section, defined by start position “Start index” and “Section size”. The tool works on gray-scale pixels values, taken from point-list parameter “Read from”. You can use previously “Point-list Read Pixels” tool to store pixel values into point-list buffer, which are then processed by this tool. Edges are detected using a binarization threshold that specifies the border value between light and dark pixels. We recommend linking the “Threshold” argument to the result of a “Percent Threshold” tool, placed on the image area of interest (where we detect edges) with approximately equal regions of light and dark pixels.

The tool detects several types of edges, specified by the “Edge direction” argument – light to dark, dark to light or both types. The “Get edge” argument specifies which edges should be stored in the result point-list - first detected edge, last detected edge, strongest edge or all edges. Combine searching of both types of edges by “Edge direction” = “Both” with “Get edge” = “All” to save all detected edges into result point-list.

The tool works on non-circular or circular input point-list as specified by the “Circular” argument.

Results:

The tool writes detected edge points into a result point-list section, starting from index “Pt-list start”. The tool returns two results – the last edge point that is stored into the result section and a float result, equal to the size of the result section (number of stored points).

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Start index of input point-list section.
Section size	Number of items in the input section.
Threshold	Edge-detection threshold value. You may use static threshold value or link the argument to a “Percent Threshold” tool.
Edge direction	Type of searched edges: “Light->Dark”, “Dark->Light” or “Both”. The last option should be used with “Get edge” = “All”.
Get edge	Specifies which types of edges are stored in the result point-list - first, last, strongest or all edges.
Circular	Specifies non-circular or circular point-list section. Set this option to “Yes” if the section contains circular data, for example circle points generated by the “Generate Point-list Circle” tool. The tool appends the beginning items at the end of the section to perform circular edge-detection.
Read from	Specifies parameter in the input point-list, which holds gray-scale pixel values – Param0 to Param 7.
Pt-list start	Start index of result point-list section.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Last edge point (P)	Last detected edge point.
Number of results (R)	Number of detected edges (size of result point-list section).

Similar tools

“Edge detection 2” tool.

Usually combined with

“Point-list Read Pixels” tool, which reads pixel values from current image into point-list.

“Percent Threshold” tool. Which specifies edge-detection threshold.

Tips & Tricks

Examples

Not indexed yet – sorry.

4.17. Point-list Distance

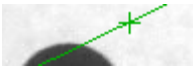
Group

Point-list tools.

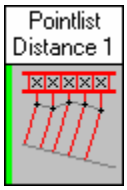
Short description

The tool calculates distances from points to a point or a line.

VIMOS kernel presentation



Editor icons



Description

General function:

The tool calculates distances between a set of points in a point-list section, defined by the arguments “Start index” and “Section size”, and a point or a line (called “distant point” and “distant line”). The “Point” argument defines the distant point. The “Point” and “Angle” arguments define the distant line - a line that crosses the point with rotation angle “Angle”. The tool finds both pixel distances and calibrated real-world distances (see argument “Find distance to”).

Results:

The tool writes calculated distances into parameter “Write distance in” of respective point-list items (Param 0, … , Param 7). The tool returns six results – two points with longest and shortest distance, their indexes in the point-list buffer and their actual distance values.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Point	Distant point.
Angle	Specifies distant line together with the distant point.
Start index	Start index of input/output point-list section.

Section size	Number of items in the input/output section.
Find distance to	Specifies type of calculated distance – to point or line in normal or calibrated (real-world) units.
Write distance in	Specifies parameter in the point-list items, which receives the distance value.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Longest dist point (P)	Point with longest distance to “distance point” or “distance line”.
Longest index (R)	Index of “Longest dist point” in the point-list buffer.
Longest distance (R)	Value of longest distance.
Shortest dist point (P)	Point with shortest distance to “distance point” or “distance line”.
Shortest index (R)	Index of “Shortest dist point” in the point-list buffer.
Shortest distance (R)	Value of shortest distance.

Similar tools

“Point-list Angle” tool.

Usually combined with

Select calibration set when calculating distances in real-world units.

Tips & Tricks

Examples

Not indexed yet – sorry.

4.18. Point-list Angle

Group

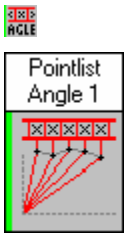
Point-list tools.

Short description

The tool calculates angles of points in a point-list section.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool calculates angles of vectors, defined by a “center” point (beginning vector point) and the points in a point-list section (end vector points). Angles are calculated according to VIMOS conventions – 0 degrees is at 12 o’clock and the angle increases clockwise.

Results:

The tool writes calculated angles into parameter “Write angle in” of respective point-list items (Param 0, … , Param 7). The tool returns six results – two points with biggest and smallest angle, their indexes in the point-list buffer and their actual angle values.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Point	Center point.
Start index	Start index of input/output point-list section.
Section size	Number of items in the input/output section.
Write angle in	Specifies parameter in the point-list items, which receives the angle value.

Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Biggest ang point (P)	Point with biggest angle.
Biggest index (R)	Index of “Biggest ang point” in the point-list buffer.
Biggest angle (R)	Value of biggest angle.
Smallest ang point (P)	Point with smallest angle.
Smallest index (R)	Index of “Smallest ang point” in the point-list buffer.
Smallest angle (R)	Value of smallest angle.

Similar tools

“Point-list Distance” tool.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.19. Point-list Best Line

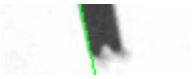
Group

Point-list tools.

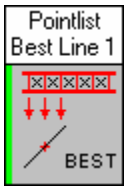
Short description

The tool generates best-fit line for points in a point-list section.

VIMOS kernel presentation



Editor icons



Description

General function:

The tool finds a line, which best fits to a set of points from a point-list section. The arguments “Start index” and “Section size” specify start section position in the point-list buffer and number of section points to process. The minimum section size is 2 points. The argument “Calculation method” specifies algorithm type - best line linear regression or Hough-line method.

Results:

The tool returns two results – one point and one angle, which define the calculated best line.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Start index of input point-list section.
Section size	Number of items in input point-list section.
Calculation method	Specifies calculation algorithm - best line or Hough line.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)

Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Best-line point (P)	Point which defines result best line.
Best-line angle (A)	Angle which defines result best line.

Similar tools

“Point-list Best Circle” tool.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.20. Point-list Best Circle

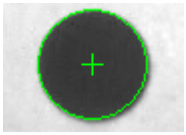
Group

Point-list tools.

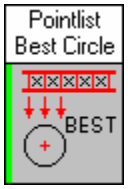
Short description

The tool generates best-fit circle for points in a point-list section.

VIMOS kernel presentation



Editor icons



Description

General function:

The tool finds a circle, which best fits to a set of points from a point-list section. The arguments “Start index” and “Section size” specify start section position in the point-list buffer and number of section points to process. The minimum section size is 3 points.

Results:

The tool returns two results – a point (center point of best circle) and a float result (radius of best circle).

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Start index of input point-list section.
Section size	Number of items in input point-list section.
Draw_clr	Tool drawing color on success (0=default : green)

Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Best-circle center (P)	Center point of result best circle.
Best-circle radius (R)	Radius of result best circle.

Similar tools

“Point-list Best Line” tool.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.21. Point-list Hough Transform

Group

Point-list tools.

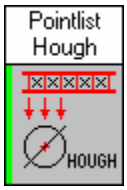
Short description

The tool approximates a Hough line or circle from a point-list.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool finds a best fit by the Hough algorithm. The tool implements two geometrical cases – line and circle. Some arguments and results have different meanings in different cases.

Input data for the tool are disconnected points placed in the input point-list section.

The main idea of the Hough transform is to represent a line or circle made of many pixels by a single peak in the parametric Hough domain or accumulator array. This peak has coordinate values in the Hough domain by two or tree parameters necessary to describe the line or circle. The tool calculates all lines and circles, which could possibly intersect each input point and accumulates all possible parameters in the Hough domain. The result will be a line or a circle with parameter's biggest peak. The tool stores best "Result num" parameter peaks into the result point-list section. The argument "Threshold" specifies minimum number of points in a detected line/circle in order to save it as a final result.

Results:

Line:

The tool returns the following results:

- Point (P) - X and Y coordinates of a point, which belongs to the best line
- Angle (A) - angle between best line and the x-axis
- Float (R) - not used in the line case
- Float (R) - number of items saved in the result point-list section

The tool stores also results in the result point-list section, sorted in descending peak order:

- Point - X and Y coordinates of a point, which belongs to a peak line

- Param 0 – angle between that line and the x-axis
- Param 1 – power of maximum (number of collected points for the line)

Circle:

The tool returns the following results:

- Point (P) – center of best circle
- Angle (A) – not used in the circle case
- Float (R) – radius of best circle
- Float (R) - number of items saved in the result point-list section

The tool stores also results in the result point-list section, sorted in descending peak order:

- Point – center of peak circle
- Param 0 – radius of peak circle
- Param 1 – power of maximum (number of collected points for the circle)

The first item in the result point-list section contains parameters of the strongest peak (best-fit line or circle).

Algorithm

Line:

The tool uses the normal equation of the line:

$$rho = x*sin(theta) + y*cos(theta),$$

where `theta` is the angle of the normal vector of the line (vector from (0,0) to nearest line point, which is perpendicular to the line), `rho` is the distance from (0,0) to the line (the length of the normal vector) and (x,y) are points from the input point-list. The `theta` and `rho` parameters take values from the parametric domain as defined by the tool arguments:

- "Min1", "Max1" and "Step1" specify minimum, maximum and step values for angle `theta` in units PI/1000 from 0 to 6283.
- "Min2", "Max2" and "Step2" specify minimum, maximum and step values for `rho`.
- "Min3", "Max3" and "Step3" are not used.

Circle:

The tool uses the circle equation:

$$(x - a)^2 + (y - b)^2 = r^2,$$

where `(a,b)` define the circle center, `r` is the circle radius and (x,y) are points from the input point-list. The parameters `a`, `b` and `r` take values from the parametric domain as defined by the tool arguments:

- "Min1", "Max1" and "Step1" specify minimum, maximum and step values for the x-coordinate of the circle center `(a)`.
- "Min2", "Max2" and "Step2" specify minimum, maximum and step values for the y-coordinate of the circle center `(b)`.
- "Min3", "Max3" and "Step3" specify minimum, maximum and step values for the circle radius `(r)`.

***ATTENTION**. The Hough transform is a computationally intensive algorithm with big memory requirements. You may reduce execution time and size of needed memory in two ways:*

- *By setting narrow parameter range [Min, Max]. Be careful when setting this range because you may miss a searched object.*
- *By setting parameter steps greater than 1. Parameter steps above 1 decrease detection precision. In general, x/y coordinate and `rho` steps in the range [2,5], as well as angle step of 8,9 (about 0.5 degrees in the line case), could be a good compromise between speed and precision.*

Another aspect of the Hough transform is the detection of numerous approximately equivalent results. The tool deletes redundant equal results in the parameter range, specified by the "step" arguments, and leaves results with the highest power of maximum. For example two circles are considered equivalent when distance between their centers is less than max(Step1,Step2) and radiuses differ no more than Step3. Greater steps reduce the number of approximately equivalent results, which could be stored in the result point-list.

Arguments

Argument	Description
Start index	Start index of input data section in the point-list buffer.
Section size	Size of input section.
Mode	Line or circle.
Min1	Minimum value for `theta` (line) or center x-coordinate (circle).
Max1	Maximum value for `theta` (line) or center x-coordinate (circle).
Step1	Step value for `theta` (line) or center y-coordinate (circle).
Min2	Minimum value for `rho` (line) or center y-coordinate (circle).
Max2	Maximum value for `rho` (line) or center y-coordinate (circle).
Step2	Step value for `rho` (line) or center y-coordinate (circle).
Min3	Minimum value for circle radius (circle only).
Max3	Maximum value for circle radius (circle only).
Step3	Step value for circle radius (circle only).
Start index	Start index of result point-list section.
Threshold	Minimum number of points in a line or a circle for a valid result.
Result num	Number of items stored in result point-list section (<= 100).

Results

Result	Description
Point (P)	A point, which defines best line or center of best circle.
Angle (R)	Angle between best line and x-axis (line only).
Float (R)	Radius of best circle (circle only).
Float (R)	Number of points saved in the result point-list section (max 100).

Similar tools

Point-list Best Line.

Point-list Best Circle.

Usually combined with

Tips & Tricks

Increase system performance by successive tool execution in coarse and fine mode. For example you can search a circle by:

- Find coarse circle radius `rad` with big radius step, for example Step3 = 10
- Find exact circle radius by Step3=1 in a narrow radius range, for example Min3=`rad-20`, Max3=`rad+20`.

Examples

Not indexed yet – sorry.

4.22. Contour Matching

Group

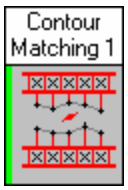
Point-list tools.

Short description

The tool compares object contours, stored in the point-list buffer and finds differences between the two objects, called ***template*** and ***target*** object. The target contour may be rotated.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool matches two object contours stored in separate point-lists. It finds a rotation angle and an error (diversion) of a target contour relative to a template contour. The tool uses as input data the distance and the angle of each contour point relative to a given center for both template and target point-lists. This data is expected to be in the parameters of each point-list item. Use “Point-list Distance” and “Point-list Angle” tools to calculate these values.

The input point-lists may consist of several sections. The first section must contain the outer contour of the template/target object. Next sections contain contours of internal objects (holes). The first item of each section should contain the following data in fixed parameters:

- Param 2: Object label (0: no label). The tool matches internal objects with equivalent labels. In case of missing matching labels or disabled label matching, the tool finds automatically corresponding internal objects, based on the distance between the object centers and the lengths of the object contours. The labeled matching is enabled by non-zero value, set in Custom parameter 1. Labels are ignored for the outer contours of the template and the target objects, which are always matched and used to find rotation angle.
- Param 3: X-coordinate of object mass center.
- Param 4: Y-coordinate of object mass center.
- Param 5: Number of points in current section.

The total point-list size is sum of section lengths.

Instructions for use:

- Place input distance and angle data into equivalent item parameters in the template and target point-lists, for example: distance in “Param 6”, angle in “Param 7”.
- Calculate distance and angle in all point-list sections, relative to the same point (entire object mass center for example).
- Don’t use reserved item parameters 2 to 5 for “**Distance from**”, “**Angle from**”, “**Error limit from**” and “**Write match in**” arguments.
- The first sections in the template and target point-lists (outer object contours) are used to find rotation angle. The tool may produce incorrect results if the outer contours are symmetrical (circle or ellipse for example) and may have more than one rotation angle.
- Currently the tool ignores the arguments **Target center** and **Template center**. They are included for compatibility with older VIMOS versions. The actual object mass centers are taken from first items in first sections of **Template Start** and **Target Start** point-lists (“Param 3” and “Param 4”).
- Currently the tool ignores the argument **Max error**. It is included for compatibility with older VIMOS versions.

Results:

The tool returns 3 results – a float result (best matching index), an angle result (rotation angle) and a float result (total accumulated error value).

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Template start	Start position of template contour in the point list buffer.
Template size	Length of template contour.
Error limit from	Specifies point-list parameter, which holds individual template-point error thresholds – Param 0 to Param 7. Should be different from 2,3,4,5. A target-point error, associated with this point, is set to 0 if the error is below the threshold. A separate toggle option disables the error threshold.
Target start	Start position of target contour in the point list buffer.
Target size	Length of target contour.
Target center	Center point of target object.
Template center	Center point of template object.
Distance from	Source of distance between center and each contour point - Param 0 to Param 7. Should be different from 2,3,4,5. Should be equal to the argument “Write result in” of the “Point-list Distance” tool, which generates the distance values. Used for both template and target point-lists.
Angle from	Source of angle (in 1/1000 radians) of each contour point relative to the center – Param 0 to Param 7. Should be different from 2,3,4,5. Should be equal to the argument “Write angle in” of the”Point-list Angle” tool, which generates the angle values. Used for both template and target point-lists.

Max error	Maximum error value, used to speed up matching (0 : disable max error speed-up)
Write match in	Target point-list parameter where the tool stores matching error for current point – Param 0 to Param 7. Should be different from the "Distance from" and "Angle from" parameters and the reserved parameters 2 to 5.
Algorithm	Matching algorithm to be used (currently PK used only).
Parameter 1	Custom parameter for the selected algorithm: • 0 : disable labeled matching of internal objects • 1 : enable labeled matching of internal objects
Parameter 2	Custom parameter for selected algorithm.
Parameter 3	Custom parameter for selected algorithm.
Beg angle	Beginning rotation angle in radians/1000.
End angle	End rotation angle in radians/1000. The two angles specify range of possible rotations, used to speed-up tool execution. Set both angles to (0,0) to disable this option.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Best index (R)	Best matching index in target point-list
Rotation angle (A)	Rotation angle of target contour relative to the template
Error (R)	Total (accumulated) matching error

Similar tools

Usually combined with

"Match Filter" tool.

Tips & Tricks

Examples

Not indexed yet – sorry.

4.23. Match Filter

Group

Point-list tools.

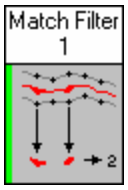
Short description

The tool performs filtering of “Contour Matching” errors.

VIMOS kernel presentation

This tool has no fixed graphical representation. It draws cross markers into overlay picture for detected error objects (areas of difference in template and target objects). Here is one example:

Editor icons



Description

General function:

The tool filters the error values stored in the point-list buffer by the “Contour Matching” tool:

- Consecutive error values >= “Match threshold” are summed into “Error objects” result.
- Consecutive error objects are merged into one, if each gap between is not longer than “Merge gap”.
- Error objects < “Object threshold” are ignored.
- Error objects shorter than “Min object length” are ignored.

This tool draws on the overlay all unfiltered error objects by placing a cross mark at each point of the error object. Larger cross marks represent larger individual error values.

Results:

The tool returns 3 float results – number of detected error objects, sum of detected error objects and biggest error object.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Start index	Point-list start index
Section size	Point-list length
Get match from	Source of error value for each point in the list: Param0 - Param7. Should be the same as 'Write match in' from Contour Matching tool.
Match threshold	Threshold to filter individual error values
Object threshold	Threshold to filter error object values
Min object length	Lower limit for error object length
Merge gap	Max gap between error objects that allows to merge them
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Error objects (R)	Number of error objects after filtering.
Error sum (R)	Sum of error objects after filtering.
Biggest error (R)	Biggest error object after filtering.

Similar tools

Usually combined with

"Contour Matching" tool.

Tips & Tricks

Examples

Not indexed yet – sorry.

4.24. Point-lists Compare

Group

Point-list tools.

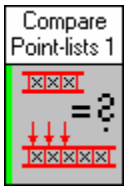
Short description

The tool finds best-matched position of one point-list in another point-list (linear and circular search).

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool compares two point-lists called ***template*** and ***target*** point-list. The template data is searched in the target point-list.

The argument "Match by" specifies which point-list parameter is matched – point X-coordinate, point Y-coordinate, Param 0 … , Param 7. The argument "Compare method" specifies searching method - absolute, relative or normalized. The absolute method looks for an absolute matching between template and target data. The relative method allows arbitrary start offset of template data in the target point-list. The normalized method searches matching of template data multiplied by some fixed coefficient value.

The target point-list is scanned in two ways – linear and cyclic. In linear mode the target point-list is scanned as linear sequence, starting from the beginning item. In cyclic mode the target point-list is treated as circular by matching beginning items after the last item.

Results:

The tool returns two float results. The first result is index in the target point-list where best match was found. The second result is match estimation in percents. It is equal to 100% if template data completely matches with target data (0% is received in case of poorly matched point-list sections).

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Template index	Start index of template point-list section.
Template size	Size of template point-list section.
Target index	Start index of target point-list section.
Target size	Size of target point-list section.
Match by	Specifies matched parameter – point x-coordinate, y-coordinate, Param 0 to Param 7.
Compare method	Specifies compare method – absolute, relative or normalized.
Scan method	Scan method – linear or cyclic.

Results

Result	Description
Best index (R)	Index of best match in target point-list.
Match (R)	Match estimation in %.

Similar tools

Fast Point-lists Compare.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.25. Fast Point-lists Compare

Group

Point-list tools.

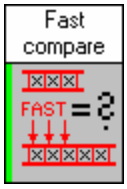
Short description

The tool finds best-matched position of one point-list in another point-list using integer arithmetic (linear and circular search). Faster version of “Point-lists Compare” tool, which uses floating-point arithmetic.

VIMOS kernel presentation

This tool has no graphical representation. It doesn’t draw into the overlay picture.

Editor icons



Description

General function:

The tool compares two point-lists called ***template*** and ***target*** point-list. The template data is searched in the target point-list. The comparison starts at target position specified by the “Start index” argument. One comparison cycle is done by incrementing template and target indexes with “Increment step”. The start target index for next comparison cycle is calculated by adding “Increment step” to current start index. The matching operation stops after “Number of steps” comparison cycles.

The argument “Match by” specifies which point-list parameter is matched – point X-coordinate, point Y-coordinate, Param 0 … , Param 7. The tool is optimized for speed and uses integer comparison only. The float parameters Param 0,…,Param 7 are converted to integer numbers with precision, specified by “Float precision”. The argument “Max error” is also aimed for speed optimization. “Max error” = 0 has no effect on comparison cycles. “Max error” != 0 aborts current comparison cycle when difference between template and target data becomes greater than “Max error”.

The argument “Compare method” specifies searching method - absolute, relative or normalized. The absolute method looks for an absolute matching between template and target data. The relative method allows arbitrary start offset of template data in the target point-list. The normalized method searches matching of template data multiplied by some fixed coefficient value.

The target point-list is scanned in two ways – linear and cyclic. In linear mode the target point-list is scanned as linear sequence, starting from the beginning item. In cyclic mode the target point-list is treated as circular by matching beginning items after the last item.

Results:

The tool returns two float results. The first result is index in the target point-list where best match was found. The second result is a match estimation, which shows difference between template and target as absolute value.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Template index	Start index of template point-list section.
Template size	Size of template point-list section.
Target index	Start index of target point-list section.
Target size	Size of target point-list section.
Start index	Start index in target point-list for first comparison cycle.
Increment step	Increment step for “Template index”, “Target index” and “Start index”.
Number of steps	Total number of comparison steps (cycles).
Match by	Specifies matched parameter – point x-coordinate, y-coordinate, Param 0 to Param 7.
Float precision	Specifies precision for float to integer conversion.
Max error	Max error value used to stop current comparison cycle when exceeded. Non-zero value increases tool speed. Zero value has no effect.
Compare method	Specifies compare method – absolute, relative or normalized.
Scan method	Scan method – linear or cyclic.

Results

Result	Description
Best index (R)	Index of best match in target point-list.
Match (R)	Match estimation as absolute value.

Similar tools

Point-lists Compare.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.26. Point-list Calculator

Group

Point-list tools.

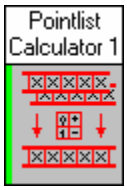
Short description

The tool performs point-list calculations.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool performs calculations with point-list parameters from 3 possible data sources – two point-list sections and one constant. The input point-lists are specified by the arguments "Source 1" and "Source 2". The tool writes result point-list starting from position "Target". All point-lists have equal size specified by "Length". The source and the target point-lists may overlap.

The two "Read from" arguments specify which "Source 1" and "Source 2" parameters participate in the calculations – Param 0 to Param 7, point x-coordinate or point y-coordinate. The "Write in" argument specifies which parameter in the target point-list receives calculation results. Arbitrary combinations of source and target parameters are possible, for example:

Target [Param 0] = Source 1 [Param 1] + Source 2 [X coord]

The "Constant" argument specifies a constant calculator operand when enabled by "Mode".

Calculator operations:

Operation	Description
+	Element by element addition: `T[k] = S1[k] + S2[k],  k=0,...,length-1` where: `S1[k]` : Source 1 point-list parameter `S2[k]` : Source 2 point-list parameter `T[k]` : Target point-list parameter `S1[k]` or `S2[k]` may be replaced by the "Constant" operand. This operation writes the target point-list. A tool result is not set.
-	Element by element subtraction:

	`T[k] = S1[k] – S2[k], k=0,...,length-1` where: `S1[k]` : Source 1 point-list parameter `S2[k]` : Source 2 point-list parameter `T[k]` : Target point-list parameter `S1[k]` or `S2[k]` may be replaced by the “Constant” operand. This operation writes the target point-list. A tool result is not set.
*	Element by element multiplication: `T[k] = S1[k] * S2[k], k=0,...,length-1` where: `S1[k]` : Source 1 point-list parameter `S2[k]` : Source 2 point-list parameter `T[k]` : Target point-list parameter `S1[k]` or `S2[k]` may be replaced by the “Constant” operand. This operation writes the target point-list. A tool result is not set.
/	Element by element division: `T[k] = S1[k] / S2[k], k=0,...,length-1` where: `S1[k]` : Source 1 point-list parameter `S2[k]` : Source 2 point-list parameter `T[k]` : Target point-list parameter `S1[k]` or `S2[k]` may be replaced by the “Constant” operand. This operation writes the target point-list. A tool result is not set.
Max Max_idx	Maximum value of “Source 1” parameters: `R = max(S1[k]), k=0,...,length-1` where: `S1[k]` : Source 1 point-list parameter `R` : Max value These operations do not write target point-list. They return as tool result the maximum value or the index of max value in the point-list buffer.
Min Min_idx	Minimum value of “Source 1” parameters: `R = min(S1[k]), k=0,...,length-1` where: `S1[k]` : Source 1 point-list parameter `R` : Min value These operations do not write target point-list. They return as tool result the minimum value or the index of min value in the point-list buffer.
Average	Average value of “Source 1” parameters: `R = (S1[0]+S1[1]+...+S1[n])/length, n=length-1` where: `S1[k]` : Source 1 point-list parameter `R` : Tool result This operation does not write target point-list and returns average value as tool result.
Variance	Variance value of “Source 1” parameters: `R = (S1[0]*S1[0]+S1[1]*S1[1]+...+S1[n]*S1[n])/length, n=length-1` where:

	S1[k] : Source 1 point-list parameter R : Tool result This operation does not write target point-list and returns variance value as tool result.

The element by element operations involve “Source 1” and “Source 2” parameters or a source point-lists and a constant. Note that some operations ignore the “Mode” argument, which specifies calculator operands, and use “Source 1” operand.

Results:

The tool writes target point-list for operations **+**, **-**, ***** and **/** (tool result is not set). The tool returns float result for operations max, min, average and variance. Although a result value is not set by some operations, the result error is set and may be checked.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Source 1	Start index of source point-list 1.
Read from	Specifies source point-list 1 parameter – Param 0 to Param 7, point x-coordinate, point y-coordinate.
Source 2	Start index of source point-list 2.
Read from	Specifies source point-list 2 parameter – Param 0 to Param 7, point x-coordinate, point y-coordinate.
Constant	Specifies floating-point constant, which participates in the calculations.
Target	Start index of result point-list.
Write in	Specifies target point-list parameter, which receives calculator results – Param 0 to Param 7, point x-coordinate, point y-coordinate.
Length	Number of point-list elements to calculate (sizes of source 1, source 2 and target point-lists).
Operation	Calculator operation (see description above).
Mode	Selects calculator operands - source 1 and source 2, source 1 and constant, source 2 and constant.

Results

Result	Description
Calculator result (R)	Result of calculator operation.

Similar tools

Calculator.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

4.27. Point-list Copy

Group

Point-list tools.

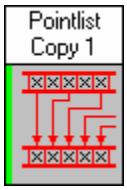
Short description

The tool copies a point-list section with optional sub-sampling (source step > 1).

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool copies a source point-list section with start address "Source start" and size "Source size" into a destination section, starting at address "Dest start". The argument "Source step" specifies source address increment. Step equal to 1 will copy the source point-list without sub-sampling. Step equal to 2 will copy each second item from the source point-list. Currently the tool applies fixed destination address increment of 1 (argument "Dest step"). In case of overlapping sections the tool may yield unpredictable results.

The argument "Source size" specifies the size of the source point-list section. The copy operation continues while current source address is less than ("Source start" + "Source size"). Example:

```
Source start = 0
Source step = 2
Source size = 10
Dest start = 20
```

The tool will copy 5 items from locations 0, 2, 4, 6, 8 into locations 20, 21, 22, 23, 24.

Results:

The tool returns one float result, equal to the number of items stored in the destination point-list.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Source start	Start index of source point-list section.
Source step	Source address increment.
Source size	Size of source point-list section.
Dest start	Start index of destination point-list section.
Dest step	Destination address increment (currently = 1).

Results

Result	Description
Dest size (R)	Size of destination point-list section.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

5. String tools

The tools in this group access the string buffer – a sequence of characters (actually bytes). The tools are used to read and write, to display or to interpret areas of the string buffer. They are useful to handle the output of ‘string generating tools’ like ‘OCR’, ‘Barcode reader’ or ‘Edit’ GUI tool, to support context sensitive help or to handle complex serial protocols.

Editor toolbar of this group of tools:

5.1. Load String

Group

String tools

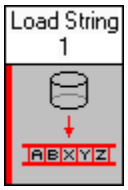
Short description

The tool loads string buffer from file.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool loads a part of the string buffer from file. The number of data bytes in the file specifies the number of loaded characters. The argument "Start" specifies start location in the string buffer where file data is stored. The argument "File" specifies source file identifier (number) – 0 for file ***sb0.vm***, 1 for ***sb1.vm*** and so on.

Note: The tool is executed in run mode only.

Results:

The tool returns two float results – number of loaded characters and index of next free cell in the string buffer.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Start	Start position in the string buffer.
File	Identifier (number) of string file ***sb***<xxx>***.vm***, where xxx is number from 0 to 999.

Results

Result	Description
Char number (R)	Number of characters loaded from file.
Next char (R)	Index of next free character in the string buffer.

Similar tools

Save String Saves a part of the string buffer to file.

Usually combined with

“Save String” tool, which creates string buffer files.

GUI tools like “Button”, which use texts stored in the string buffer.

Tips & Tricks

There is nothing special to say about this tool.

Examples

Not indexed yet – sorry.

5.2. Save String

Group

String tools

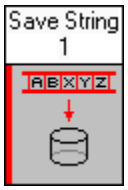
Short description

The tool saves string buffer to file.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool saves a part of the string buffer to file. The arguments "Start" and "Length" specify start position in the string buffer and number of characters to save. The argument "File" specifies destination file identifier (number) – 0 for file ***sb0.vm***, 1 for ***sb1.vm*** and so on.

ATTENTION*. This tool writes a file, which may quickly consume all available flash space on the camera.*

Note: The tool is executed in run mode only.

Results:

The tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Start	Start position in the string buffer.

Length	Number of characters to save
File	Identifier (number) of string file ***sb***<xxx>***.vm***, where xxx is number from 0 to 999.

Results

The tool has no results.

Similar tools

Load String Loads a part of the string buffer from file.

Usually combined with

“Load String” tool, which loads files to string buffer.

Tips & Tricks

There is nothing special to say about this tool.

Examples

Not indexed yet – sorry.

5.3. Put Char

Group

String tools

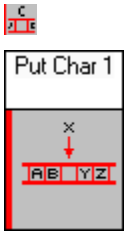
Short description

The tool puts a character into a given cell of the string buffer.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool writes a character at a specific location in the string buffer:

```
string_buffer[<start> + <offset>] = <char>
```

Note: The tool is executed in run mode only.

Results:

This tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Char	ASCII code of the character (see "APPENDIX A. ASCII table").
Start	Start position in the string buffer.
Offset	Offset from the start position.

Results

This tool has no results.

Similar tools

Load string buffer	loads a part of the string buffer from flash file
Number to string	converts a number to digits and writes the result to a part of the string buffer
OCR	writes the result of an optical character recognition to the string buffer
Barcode	writes the result of barcode reading to the string buffer
Get Char	reverse operation – reads the content of a cell of the string buffer

Usually combined with

No typical combination comes to mind at the moment ☺.

Tips & Tricks

There is nothing special to say about this tool.

Examples

Not indexed yet – sorry.

5.4. Get Char

Group

String tools

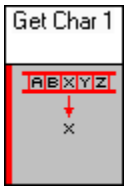
Short description

The tool fetches a character from a given cell of the string buffer

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

This tool retrieves a character from a specific location in the string buffer:

```
<char> = string_buffer[<start> + <offset>]
```

Results:

The tool returns the contents of the addressed cell of the string buffer, usually an ASCII code (see "APPENDIX A. ASCII table").

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Start	Start position in the string buffer.
Offset	Offset from the start position.

Results

Results	Description
Character (R)	ASCII code of the character (see "APPENDIX A. ASCII table").

Similar tools

Show string	displays the contents of a part of the string buffer
Put Char	reverse operation – writes the content of a cell of the string buffer

Usually combined with

No typical combination comes to mind at the moment ☺.

Tips & Tricks

There is nothing special to say about this tool.

Examples

Not indexed yet – sorry.

5.5. Number to String

Group

String tools

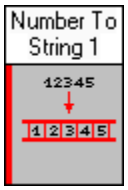
Short description

This tool converts a value or tool result into a string and stores the string in the string buffer.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool converts a number to string (digits) and stores the string in the string buffer. The argument "Number" specifies a direct float value or a tool result when linked. The argument "Type" specifies type of output string. For decimal, hex and binary types the tool rounds the float number to nearest integer and then converts the integer to string in specified radix.

The last 5 tool arguments ("Total length", "Fraction type", …) specify total length of result string (auto or fixed), left/right alignment with padding, and also fraction length and rounding for the 'float' type. You can specify automatic or fixed length of the result string and the fraction. When using fixed string length you may align converted number left or right in the result string with right or left padding of zeros and blank characters.

***ATTENTION**. The formatting options, specified by the last 5 tool arguments, are not available on ADSP camera (TI camera and PC simulator only).*

.

Integer formatting

The following combinations of "Total length" and "Fraction length" specify formatting for the integer (decimal/hex/bin) type:

Total length	Fraction length	Description
Auto	**-**	Automatic length of result string, no alignment, no char padding from left or right.

Fixed	**0**	Fixed length of result string with **left** alignment of the integer number. Right padding with specified char – '0' (space used), space or 0x00 (string truncation).
Fixed	**>0**	Fixed length of result string with **right** alignment of the integer number. Left padding with specified char – space (default) or '0'.

Float formatting

The following combinations of "Total length" and "Fraction type" specify formatting for the 'float' type:

Total length	Fraction type	Description
Auto	**Auto**	Automatic length of result string, automatic fraction length, no alignment and char padding.
Auto	**Fixed length** **Fixed length & round**	Automatic length of result string, fixed fraction length (specified by "Fraction length") without/with rounding, no alignment and char padding.
Fixed	**Auto**	Fixed length of result string, automatic fraction length. **Left** alignment of the float number in the result string. Right padding with specified char – '0' (default), space or 0x00 (string truncation).
Fixed	**Fixed length** **Fixed length & round**	Fixed length of result string, fixed fraction length (specified by "Fraction length") without/with rounding. **Right** alignment of the float number in the result string. Left padding with specified char – space (default) or '0'.

Results:

The tool returns one float result – the length of the string stored in string buffer. The tool stores the result string without a terminating 0-char. If the total string length is fixed and insufficient to hold all significant non-fractional digits of the number, the tool truncates the least significant digits of the number (from right) and returns error 9004.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Number	Number to convert – a direct float value or linked to some result.
Type	Type of result string: • **Dec** – decimal • **Hex** – hexadecimal (hex digits 0-9,A-F) • **Bin** – binary (sequence of '0' and '1' chars). • **Float** – floating-point (default)
Start position	Start position of result string in the string buffer.
Total length	Total length of result string – automatic or fixed (1 to 32).
Fraction type	Fraction type – automatic length, fixed length without rounding, fixed length with rounding (used for float type only).

Fraction length	Fraction length used when: • "Fraction type" is not auto for float type. • For integer types (Dec, Hex and Bin) – specifies left or right alignment (see tool description).
Left fill	Specifies left padding char when the result string has fixed length and is aligned right – space or '0' (see tool description).
Right fill	Specifies right padding char when the result string has fixed length and is aligned left – '0', space or 0x00 (see tool description). The last padding char actually truncates the result string.

Results

Results	Description
Length (R)	Length of result string.

Similar tools

String to Number reverse operation – converts a sequence of digits to value (result)

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

5.6. String to Number

Group

String tools

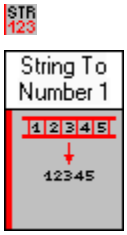
Short description

This tool converts a sequence of numerals (digits) from the string buffer into a number.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

This tool converts a string from the string buffer into a number. You should specify a starting position of string and tell the tool what kind of number it is (decimal, hexadecimal, binary or floating point). Specify also a maximum string length. The tool seeks to interpret the data as a number and returns its value.

Results:

The tool returns a result that takes the value of the 'decoded' number and another result which points at the location in the string buffer behind that number.

Algorithm

The tool 'walks' over a sequence of chars in the string buffer beginning from the start location. If there are codes that do not 'decode' into valid digits for the given type of number, the tool stops number generation at that point and returns the number generated up to that moment. For instance the letter 'A' would not be valid for most types of numbers, but for hexadecimal numbers it's OK. Binary numbers may contain only '0' and '1'. Please note, that '0' in that case means the ASCII code for the numerical which is hexadecimal 0x30 or decimal 48 (see "APPENDIX A. ASCII table"). The value 0x00 or decimal 0 (zero) is always 'wrong' for any kind of number, so that the string interpretation ends always if a zero is found.

Arguments

Argument	Description

Start	Start of string in the string buffer.
Length	Max string length. The string ends at the first null character.
Null term	Currently ignored. The string is always truncated by a null inside first “Length” characters (see “Length” argument).
Type	Type of result string: Dec - decimal; Hex - hexadecimal; Bin - binary; Float - floating-point.

Results

Results	Description
Number (R)	Converted number.
Position (R)	Next position in the string buffer behind the input 0-term string.

Similar tools

Number to string	reverse operation – converts a number and writes its digits to the string buffer

Usually combined with

OCR	writes the result of an optical character recognition to the string buffer
Barcode	writes the result of barcode reading to the string buffer

Tips & Tricks

The tool has mainly two intentions. The first is to enable the use of strings for control of all the other tools. So now you can compare the result of a barcode reading to a numerical range given by the user. Or you can read a number via OCR and then use the result to call a given program that handles exactly the kind of product identified by this number.

The second purpose of this tool is a terminal like communication. The digits entered by the user and written into the string buffer can now be translated into real numerical values.

Examples

Not indexed yet – sorry.

5.7. Show String

Group

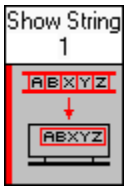
String tools

Short description

The tool displays a string on the screen.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool displays a string enclosed in a box in the overlay. The string is read from the string buffer, starting from location “Start” with maximum length “Length’. The string is truncated if a null character (code 0) occurs in first “Length’ characters. The size of the string box is determined by two arguments - “Width” specifies box width and “Text size” specifies small or big text font.

Results:

This tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Point	Point which defines position (x,y) of string box on screen.
Width	Box width in pixels, 0 = size to fit.
Start	Start of string in the String buffer.
Length	Max. string length. The string ends at first null character (code 0).

Null term	Currently ignored.
Text Size	Font size used to display the string- 1=small, 1=big.
Align	Horizontal text alignment within the box – left, centered or right. Used when "Width" specifies non-zero box width, which is greater than actual string length in pixels.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

This tool has no results.

Similar tools

"Text box" tool - displays text and numbers, but texts are read from the tool's data in the user-program.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

5.8. Compare String

Group

String tools.

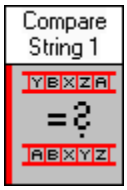
Short description

Compares two areas in the string buffer.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icon



Description

General function:

The tool compares two strings in the string buffer or a dynamic string from string buffer and a static string, read from the user program. The arguments "Start 1", "Length 1", "Start 2" and "Length 2" define start positions and lengths of two dynamic strings in the string buffer. The "Static string" argument specifies string, which is read from the user-program and can be initialized by Editor only. The dynamic string operands are truncated if a null character (code 0) occurs in first "Length 1" or "Length 2" characters. The static string is always null-terminated. "Null-term" is currently ignored. The argument "Case sensitive" specifies whether the compare operation is a case-sensitive or not. The "Mode" argument selects strings to compare – two dynamic strings or dynamic (string 1) and static string.

Results:

The tool returns one float result, which is set according to the results of the comparison operation:

- 0 – different strings
- 1 – one string is contained in the other, starting from the beginning
- 2 – equal strings.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description

Start 1	Start position of first string in the string buffer.
Length 1	Max length of first string. The string ends at the first null character.
Null-term	Use null-terminated strings (currently ignored, see description).
Case sensitive	Case sensitive comparison: Y - case sensitive; N - ignore case
Static string	Static string entered by Editor. Used when “Mode” = “Str2 & Static”.
Start 2	Start position of second string in the string buffer.
Length 2	Max length of second string. The string ends at the first null character.
Mode	Selects strings to compare: • **Str1 & Str2.** Compares the dynamic strings – string1 and string 2. • **Str1 & Static.** Compares string1 and the static string.

Results

Results	Description
Comp. result (R)	Compare result: • 0 - strings are different • 1 - one string is a prefix of the other (the shorter string coincides with the beginning characters of the longer string) • 2 - strings are equal

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

6. GUI tools

The tools in this group communicate with the user-program during its execution. GUI tools accept user input (mouse commands or keyboard codes) in a “GUI” mode – a special run mode, in which mouse cursor is visible on the screen. You cam use the mouse to click buttons, change spin values, check radio-buttons, etc. VIMOS supports other pointing devices as well – touch-screen for example. GUI tools work in PC Simulator and on TI camera. You are able to enter text on the camera by a virtual keyboard via the mouse. On PC a virtual keyboard is not needed and not supported – you enter text by the PC keyboard.

Editor toolbar of this group of tools:

6.1. Working with GUI tools

6.1.1. GUI texts

You can specify GUI tool texts (button label for example) in two ways:

- By texts from the ***string buffer*** – a special VIMOS memory buffer for character data. The “start” tool arguments usually specify beginning positions of texts in the string buffer. In general tool strings are 0-terminated or truncated to given length (if 0-char is not found in first “length” characters). Some tools use a sequence of 0-terminated strings.
- By direct string arguments (the “Label” arguments), entered by the Editor. This is the default method available in VIMOS versions 2.55 and higher. The maximum length of the “Label” string arguments is 64 chars. If you need longer strings, you should use the string buffer. Some tools need a sequence of strings - the spin tool (type = toggle) and the radio-button menu. Use the ‘**~**’ character to terminate separate strings in the “Label” argument, for example:

  Label = **“**`Option 1~Option 2~Option 3`**”**

  The total length of all ‘~’-separated strings should not exceed 64 chars.

When using the first method you must load respective data in the string buffer. You must generate a VIMOS string file and load the file into the string buffer. This can be done dynamically by the “**Load string**” tool or you can specify default string file, which is loaded on VIMOS start. Use the following options in the “**General purpose configuration**” dialog to specify default string file:

- **Load string file =** `sb0.vm` to `sb29.vm`. Name of default file, which is loaded into string buffer.
- **DRAM buffer clear =** Off. Disable clearing of DRAM buffers (point-list and string buffer) when loading new user program.

The system configuration settings are saved when you exit the dialog by pressing the OK button. Copy the necessary string file(s) into the workspace folder of the VIMOS Simulator before you run VIMOS kernel on Simulator. Load string file(s) to camera before you run VIMOS kernel on camera.

String files can be created by VIMOS kernel (the “**Save string**” tool) or by the string compiler (see next section).

ATTENTION*. When you use the first method you load new data into the string buffer, you will see new GUI texts when the kernel performs one execution pass of the user program. This pass is done automatically in edit mode when you configure some tool and press OK button in its configuration dialog or when you run or edit user-program*

*from editor in simulator. But you won't see new strings just when you exit the "**General purpose configuration**" dialog.*

6.1.2. String compiler

The string compiler STRC.EXE is a Windows console program, which generates VIMOS string files. The program reads standard text file (CR, LF-terminated text lines), created by most text editors. Each row in the input file is converted to a 0-term string and written into the output string file. Provide dummy text rows if you need to reserve some space in the string buffer. The program generates second output text file with the start string positions of the texts in the string buffer, relative to 0 (the beginning of the string buffer).

Example:

STRC *str0.txt sb0.vm str0.pos*

Where:

str0.txt = input text file

sb0.vm = output VIMOS string file

str0.pos = output string-position text file (default name ***strc.pos*** when missing)

Read ***str0.pos*** to set correct start string positions when configuring GUI tools.

6.1.3. GUI tools in edit mode

GUI tools are organized in groups. Each GUI tool has an argument, which specifies the group the tools belong to. The GUI tools from the active group are displayed in the edit mode of the VIMOS kernel, while the GUI tools from the other groups are hidden. The active group may be set by the "**GUI group id**" option in the "**General-purpose configuration**" dialog. The GUI group is automatically changed when GUI tools are selected consecutively by the user-program browser – the current selected GUI tool is always visible. Using of groups is very useful when you edit GUI tools, which occupy the same screen area, but are executed in different branches of the user-program. By default all GUI tools belong to group 0.

***ATTENTION**. GUI tools and normal tools may overlap (GUI tools override non-GUI tools). All GUI tools become invisible (their groups become inactive), when you select a non-GUI tool.*

6.1.4. GUI tools in run mode

The GUI tools, which are executed in the current user-program pass, are displayed on the screen (the "group id" argument applies to edit mode only). If you want to hide some GUI tools, place them in a user-program branch, which is not executed. Use IF-ELSE-ENDIF, GOTO and LABEL constructions. Thus, depending on run-time conditions, you are able to display and work with different sets of GUI tools, which may occupy the same screen area.

Interaction with GUI tools by mouse commands is possible in GUI run mode, when the mouse cursor is displayed on the screen. When you start a user program the GUI mode is disabled. Use the "**GUI setup**" tool or left mouse click to enter GUI run mode. Use the "**GUI setup**" tool to exit GUI run mode. Thus, depending on certain conditions, you may enable or disable access to GUI tools, which configure different parameters of your application.

***ATTENTION**. Proper execution of GUI tools (reaction to user commands and respective redrawing) is possible when each pass of the user-program goes through all modules in the main system loop:*

- *I/O module - gets mouse/touch-screen commands.*
- *UPX module – executes one pass of the user program.*
- *DRM module – redraws tools including changed GUI elements like pressed buttons.*

You should not make internal GOTO-LABEL loop in the user program, which waits for a pressed button for example, because the I/O module and the DRM modules are not executed. Below is shown example of user-program loop, which hangs the system.

Example of user-program loop, which hangs the system:

```
........
LABEL0
Button [R1]
IF(R1==3) GOTO LABEL1      /* Exit GOTO-LABEL0 loop on button click */
GOTO LABEL0
........
LABEL1
```

6.1.4.1. Relationship between tool arguments and results

Some GUI tools have input arguments, which specify initial tool results – for example the spin value. Such arguments can be linked to results of other tools, which change from time to time. At the same time user commands may change the tool results (the spin value in our example). This section explains how a tool result is generated, when it can be set from a linked argument and by a user command.

There is a common rule - GUI tools keep track of changes in argument values. Tools are updated (redrawn) and new result values are generated in both cases:

- when values of linked arguments change;
- when mouse commands change tool appearance or result.

So if you have set a given spin value by a mouse command, the spin result will remain unchanged while the “Spin value” argument is not altered.

You may force updating of all GUI tool results from respective input arguments by the "GUI setup" tool (**Setup GUI tools** = On). In general tool results receive default values from input arguments in the following cases:

- When the user-program is loaded and executed for the first time.
- When a tool configuration dialog is closed by the OK button in edit mode.
- When an argument is linked to a result and its value is changed in run mode.

In all other cases the tools keep the user-specified results.

6.1.5. Mouse commands

Use the left mouse button to work with GUI tools. Remember that right mouse click is reserved for opening the run main menu, which stops run mode and enters edit mode. The “**GUI setup**” tool may disable opening of the run main menu.

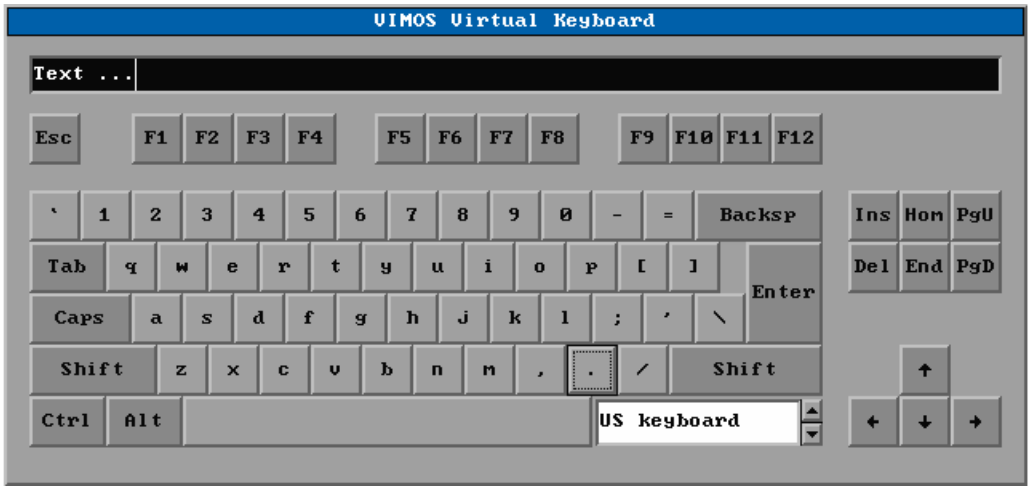
6.1.6. Virtual keyboard

Some GUI tools (edit, spin) open virtual keyboard for entering text and/or numbers. The virtual keyboard is opened by a left mouse click on the tool's edit box when the "edit" operation has been enabled by the tool's configuration. The virtual keyboard window contains buttons, simulating the buttons of a normal keyboard without the numeric pad, an edit box, which displays the entered text, and a spin control, which selects international keyboard layout.

The Simulator does not support the virtual keyboard. Select a GUI tool by a left click on the tool's box and then use the PC keyboard to enter text. The editing rules are the same as if you are entering text via the virtual keyboard, but you don't need to press Enter to terminate the edit operation.

***ATTENTION**. You will see an initial text in the edit box when you open the virtual keyboard. The initial text will be preserved if the first pressed button is one of the following buttons - **Backsp**, **Ins**, **Del**, **Home**, **End**, **Cursor right**, **Cursor left**. The initial text will be deleted if the first pressed button is an ASCII one (not a functional key). Press **Esc** to restore the initial text at any moment of the editing process.*

Virtual keyboard window:



Enter text by pressing ASCII buttons (see "APPENDIX A. ASCII table"). Use functional buttons to edit text (described below). Note that most keyboard buttons accept continuous pressing for fast button operation. Press the "Enter" button to close the virtual keyboard window. The keyboard text will be returned to the GUI tool, which has invoked the keyboard.

Usage of functional (gray) keyboard buttons:

Functional button	Function
Esc	Discards changes and restores initial keyboard text.
Backsp	Deletes character in front of cursor and moves cursor left.
Enter	Closes virtual keyboard.
Caps	Toggles capital letters on/off. Button letters are changed depending on the "Caps" state. A second press on "Caps" resets its state.
Shift	Enters next pressed key in "Shift" state. A second press on "Shift" resets its state.
Ins	Toggles insert mode on/off and changes cursor shape.
Del	Deletes character at cursor position.
Home	Moves cursor to the beginning of the text.

End	Moves cursor to the end of the text.
Cursor right	Moves cursor one position right.
Cursor left	Moves cursor one position left.
Up/Down spin buttons	Change keyboard type.

Note: All functional buttons not described in the table above are reserved for future VIMOS versions.

6.1.7. Touch-screen

The VIMOS kernel on TI camera accepts commands from a touch-screen device, based on the AHL-51S “Analog Touch Panel Controller”. The touch-screen must perform serial communication at TTL levels (+/-5 V) and should be connected to the keypad port of the TI camera. The touch-screen must be configured in “Make and break” mode. GUI tools recognize simultaneous input from mouse and touch-screen (higher priority has the mouse).

6.1.7.1. Limitations

Compared to mouse, working with touch-screen has some limitations. For example you can’t drag objects like GUI rectangles. The possible touch-screen operations are:

- Press touch-screen spot – analogous to left mouse button press (transition from released to pressed state).
- Keep pressed touch-screen spot - analogous to keeping left mouse button in pressed state. Moving of pressed touch-screen spot is ignored.
- Release touch-screen spot - analogous to left mouse button release (transition from pressed to released state).

6.1.7.2. Calibration

Enter **Edit main menu > Configuration > Calibrate touch-screen**. Press two touch-screen spots in top left and bottom right screen corners, marked by small cross markers on the monitor. Right mouse click aborts calibration and returns to parent menu. Calibration parameters are saved into system initialization file on exit, so you don’t need to make calibration on each kernel start.

6.2. GUI setup

Group

GUI tools.

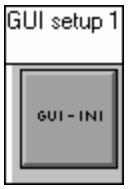
Short description

The GUI setup tool performs various setup operations, necessary to work with GUI tools.

VIMOS kernel presentation

None

Editor icons



Description

The GUI setup tool performs the following operations:

- Enters or exits GUI run mode – the mode in which you are able to work with GUI tools. In GUI mode you will see mouse cursor on the screen.
- Initializes (redraws GUI tools, which is necessary when you change the font by loading a new font file.
- Loads new font file, which changes font of GUI tools. Must be used in combination with item 2.
- Enables or disables opening of run main menu, which may be used to stop run mode and enter edit mode. Use this option to disable edit mode of VIMOS kernel for an end user.

Results :

The tool does not return results.

Arguments

Argument	Description
Set GUI mode	Change type of run mode. Use on of the following options: • **No change**. Don't change type of current run mode. • **On**. Enter GUI mode to work with GUI tools (the mouse cursor appears). • **Off**. Exit GUI mode to disable interaction with GUI tools (the mouse cursor disappears)
Setup GUI tools	Reinitialize (redraw) all GUI tools in current user program:

	• **No**. Don't initialize GUI tools. • **Yes**. Initialize GUI tools. Use this option to redraw GUI tools when changing GUI font (see next option).
Load font file	Load font file (currently not supported). Use the following options: • **None**. Don't load font file. • **`fnt0.vm`** to **`fnt29.vm`**. Name of font file (enabled by "Setup GUI tools' = Yes).
Edit mode	Set access to edit mode of VIMOS kernel. Options: • **Off**. Disable opening of run main menu. • **On**. Enable opening of run main menu by right mouse click to stop run mode and to enter edit mode.

Similar tools

None.

Usually combined with

Other GUI tools. Used to enable or disable GUI mode.

Tips & Tricks

Examples

Not indexed yet – sorry.

6.3. Window

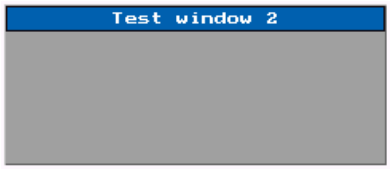
Group

GUI tools.

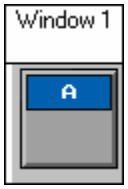
Short description

The tool displays a window on the screen.

VIMOS kernel presentation



Editor icons



Description

General function:

The window tool provides workplace where to put other GUI tools. It does not react to mouse commands and has no results - it just draws a window on the screen. The window can be moved in the edit mode only.

Results:

None.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Point	Upper left window corner.
Title start	Start position of window title in the string buffer, used when “Mode” = string buffer.
Title length	Length of title string. The tool accepts 0-term strings with length

	less or equal to “Title length”. If 0-term character is not present in first “Title length” characters, starting from “Title start”, then exactly “Title length” characters are read from string buffer for window title.
Width	Window width in pixels.
Height	Window height in pixels.
Depth	Window depth in pixels (from 1 to 3).
Title bgnd color	Background color of window title. Select one of the following options: • **Default**. Use default color • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Title fgnd color	Foreground color of window title. Select one of the following options: • **Default**. Use default color • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Mode	GUI text mode (window title source) – string buffer or “Label” argument.
Label	Direct string argument used when “Mode” = label.
Group id	Number of GUI group, the tool belongs to. This option is used in edit mode only. Place GUI tools, which overlap on the screen, in different groups.

Similar tools

Other GUI tools.

Usually combined with

Other GUI tools.

Tips & Tricks

Non-GUI tool drawings are hidden behind the window. The window tool should precede all GUI tools, which are displayed over it.

You may disable window caption (the title box) by setting “Title bgnd color” to “Gray” and by using empty (NULL) title string.

Examples

Not indexed yet – sorry.

6.4. Button

Group

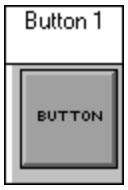
GUI tools.

Short description

The tool displays a button on the screen.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool draws a 3-D button on the screen. Press and release the button by the mouse. Depending on its current state and the mouse command, the button tool generates different results. The button picture is a text, read from the string buffer or the “Label” argument.

Results:

The button tool has one result, which receives current button state or press/release event (see **Results** table).

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Button point	Upper left button corner.
Type	Button type: • **Text**. Button text taken from the string buffer or the “Label” argument. • **Icon**. Currently not supported, will be implemented in future VIMOS releases.

Text start	Start position of button text in the string buffer, used when “Mode” = string buffer.
Text length	Length of button text if not 0-term. The tool accepts 0-term strings with length less or equal to “Text length”. If 0-term character is not present in first “Text length” characters, starting from “Text start”, then exactly “Text length” characters are read from string buffer for button text.
Align	Horizontal alignment of button text: • **Center**. Align button text to center (default) • **Left**. Align button text left. • **Right**. Align button text right. Note: Button text is always centered vertically.
Width	Button width in pixels.
Height	Button height in pixels.
Depth	Button depth in pixels (from 1 to 3).
Bgnd color	Background button color: • **Default**. Use default color (gray) • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Fgnd color	Foreground color of button text: • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Mode	GUI text mode (button text source) – string buffer or “Label” argument.
Label	Direct string argument used when “Mode” = label.
Group id	Number of GUI group, the tool belongs to. This option is used in edit mode only. Place GUI tools, which overlap on the screen in different groups.

Results

Result	Description
Button event	Button press/release event or state: • 0 : the button is in released state (no button event) • 1 : button press event (transition from released to pressed state) • 2 : the button is in pressed state • 3 : button release event (transition from pressed to released state)

Similar tools

Other GUI tools.

Usually combined with

Other GUI tools.

“Load string” tool – can be used to load texts from external file into the string buffer.

Tips & Tricks

You can change dynamically button appearance in run mode by linking button arguments “Start” and “Color” to altering results of other tools.

Examples

Not indexed yet – sorry.

6.5. Edit

Group

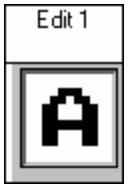
GUI tools.

Short description

The tool displays a box with text. Change the text by the virtual keyboard on TI camera or the PC keyboard in Simulator. The tool has options for text display without editing.

VIMOS kernel presentation

Editor icons



Description

General function:

The edit tool displays a text, enclosed in an optional box. The text can be taken from several sources, specified by the "**Mode**" argument:

- The string buffer. A 0-terminated string is read from the string buffer starting from position "**String start**". The "**String length**" argument specifies initial length of the text (if the 0-term character is not present in first "**String length**" characters). A second length parameter "**Max edit length**" specifies max number of characters allowed to enter for the "Edit" tool type. This is the only option, which can be used in edit mode ("**Type**" = "Edit"). For show/text types you can use both text sources.
- The "**Label**" argument. This is the default option for text source in display mode ("**Type**" = "Show" or "Text"). This option is ignored for the "Edit" tool type.
- The "**Value**" argument is converted to string and displayed. Can be used if "**Type**" = "Show" or "Text", ignored for the "Edit" tool type.

There are 3 tool types:

- **Edit**. The tool shows a text from the string buffer, which may be changed. On PC you can enter text by the PC keyboard. On TI camera you can enter text by the virtual keyboard, opened by left click on the edit box (see "6*.1.6. Virtual keyboard*"). Enter the text and close the virtual keyboard by the "Enter" button. The entered text is saved as a 0-term string in the string buffer, starting from "**String start**". Remember that the changed edit string may have different length, compared to the length of the initial string. The maximum allowable length of the edit string is specified by the "**Max edit length**" argument, which means that **("Max edit length" + 1)** characters could be stored in the string buffer. This option **ignores** the arguments "**Align**", "**Mode**", "**Value**" and "**Label**" and edits text from the string buffer, left aligned in the edit box.

- **Show**. The tool shows a text, enclosed in box, but you can't change the text. The text can be aligned left (default), centered or aligned right as directed by the "**Align**" argument. The text is always centered vertically in the box. The text is taken from the string buffer, from the "**Label**" argument or from the "**Value**" argument (see the "**Mode**" argument).
- **Text**. Same as the "Show" type, but the text is not enclosed in a box.

The last two types can be used to display various data in a GUI window.

***ATTENTION**. In edit mode you have to load string buffer with initial text if necessary. Remember to ensure "Max edit length" + 1 characters free string buffer space for the edited text.*

Results:

The edit tool returns 3 float results – edit event, start position and length of edit string

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Edit point	Upper left corner of edit box
Type	Type of edit tool: • **Edit**. Displays a text, which may be changed by the virtual or the PC keyboard. • **Show**. Displays a text enclosed in box (editing is not possible). • **Text**. Displays a text without an enclosing box (editing is not possible). The text foreground color is specified by **"Unsel fgnd color**". The text is displayed with default window background color, centered vertically in the edit area.
String start	Start position of edit text in the string buffer.
String length	Length of the edit string when not 0-terminated. In edit mode this argument is used to truncate the string in the first execution pass of the user-program after it has been loaded into VIMOS. In run mode this argument truncates the string when the argument value is changed (if linked) or when the string is changed. **Note**: This argument is also changed when the keyboard modifies the string. If the argument has been linked, it will be unlinked.
Max edit length	Maximum length of the edit string – max number of characters, which may entered by the keyboard. Since the tool writes a 0-term string in string buffer, ("**Max edit length**" + 1) characters may be stored into the string buffer, starting from "**String start**".
Width	Width of edit box in pixels.
Height	Height of edit box in pixels.
Depth	Depth of edit box in pixels (from 0 to 3).
Unsel bgnd color	Background color of unselected edit tool (not on focus):

	• **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Unsel fgnd color	Foreground color of unselected edit tool (not on focus): • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Sel bgnd color	Background color of selected edit tool (on focus): • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Sel fgnd color	Foreground color of selected edit tool (on focus): • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Align	Horizontal text alignment in the edit box for tool types “Show” and “Text” – left, center or right. Ignored for tool type “Edit”.
Mode	GUI text mode (text source) – the string buffer, the “**Label**” argument or the “**Value**” argument. Forced to string buffer for tool type “Edit”.
Value	Float number, which is converted to string when “**Mode**” = value. Ignored for tool type “Edit”.
Label	Direct string argument used when “**Mode**” = label. Ignored for tool type “Edit”.
Group id	Number of GUI group, the tool belongs to. This option allows configuring of GUI tools in edit mode of VIMOS kernel. We recommend putting overlapping GUI tools in different groups.

Results

Result	Description
Edit event	Edit event: • 0 : no edit event • 1 : the tool is put on focus by mouse click on the edit box • 2 : edit string changed
Start position	Start position of the edit text in the string buffer (equal to the “String start” argument).
String length	Length of current 0-terminated edit string in the string buffer, excluding the 0-term character.

Results

Result	Description
Edit event	Edit event: • 0 : no edit event • 1 : edit tool put on focus by click on edit box • 2 : edit string changed
Start position	Start position of the edit text in the string buffer (equal to the “String start” argument).
String length	Length of current 0-terminated edit string in the string buffer, excluding the 0-term character.

Similar tools

Other GUI tools.

Usually combined with

Other GUI tools.

Load GUI tool texts from external file into the string buffer by the “Load string” tool.

Tips & Tricks

Examples

Not indexed yet – sorry.

6.6. Spin

Group

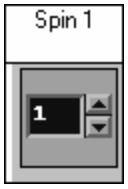
GUI tools.

Short description

The tool displays a box with value and two buttons, which increment or decrement the value. Use PC or virtual keyboard to enter new value by left click inside spin box (if editing is enabled). The “toggle” option of the tool displays texts instead of numbers.

VIMOS kernel presentation

Editor icons



Description

General function:

The spin tool displays integer or floating point value, enclosed in a box. Two buttons (called “Up” and “Down” button) increment or decrement current spin value by a given step. The tool returns as result the current spin value. The tool supports 3 types of spin values:

- Integer (“**Spin type**” = “Int “, “Int & edit”). The tool displays an integer number. Input float arguments are rounded to nearest integer value.
- Float (“**Spin type**” = “Float “, “Float & edit”). The tool displays a floating-point number with integer part, decimal point and fraction.
- Toggle (“**Spin type**” = “Toggle”). The tool displays texts instead of numbers,

The height of the spin buttons is determined by the height of the spin box. The width of the buttons is variable (see description of the argument “**Max edit length**”.

Integer and float tool types:

The arguments “**Spin value**”, “**Spin step**”, “**Min spin value**”, “**Max spin value**” and “**Max edit length**“ are used for the integer and float tool types. The toggle arguments are ignored. For tool types “Int & edit” or “Float & edit” you can enter direct spin values by the virtual keyboard on TI camera or the PC keyboard in Simulator. Start an edit operation by a left click on the spin box. Close the virtual keyboard by the “Enter” key (see “*6.1.6. Virtual keyboard*”). The “**Max edit length**” argument specifies the maximum number of digits allowed to enter.

Toggle tool type:

The arguments "**Toggle start**", "**Toggle count**" and "**Toggle state**" are used for the toggle tool type. The spin arguments, valid for the integer and float types, are ignored. The toggle tool type displays sequentially "**Toggle count**" texts, read from the string buffer or from the "**Label**" argument (see the "**Mode**" argument). In "String buffer" mode the tool reads a sequence of 0-terminated strings from the string buffer, starting from "**Toggle start**". In "Label" mode the tool reads a sequence of '**~**'-terminated strings from the "**Label**" argument. An example for 2 toggle options:

Label = "`Toggle option 1~Toggle option 2`"

The total length of all strings in "**Label**" cannot exceed 64. If you need longer texts, use the "String buffer" option.

The up/down spin buttons select previous/next toggle text. In toggle mode the tool returns as result the current toggle state, equal to the index of the selected string in the range [0, "**Toggle count**"-1]. You can't change toggle strings by the keyboard.

Results:

The spin tool returns 2 float results – spin event and current spin value or toggle state..

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Spin point	Upper left corner of spin box
Spin type	Type of spin tool: • **Int**. Integer spin value. Virtual keyboard is disabled. • **Int & edit**. Integer spin value. Virtual keyboard is enabled. • **Float**. Float spin value. Virtual keyboard is disabled. • **Float & edit**. Float spin value. Virtual keyboard is enabled. • **Toggle**. Select one toggle state among several toggle states. Virtual keyboard is disabled.
Spin value	Initial spin value. This value becomes current "spin value" result in the following cases: • When the user-program is loaded (and executed) first time. • When the spin configuration dialog is closed by the OK button in edit mode. • When the argument is linked to result and its value is changed in run mode. See section "6.*1.4.1. Relationship between tool arguments and results*".
Spin step	Spin step, used to increment/decrement current spin value by "Up" and "Down" buttons.
Min spin value	Minimum spin value, which can be set by the "Down" button.
Max spin value	Maximum spin value, which can be set by the "Up" button.
Max edit length	Maximum length of keyboard string entered for types "Int & edit" and "Float & edit". Also specifies the actual width of the spin box in character number. If this width is less than default box width, equal to the total tool width "Width" minus the minimum button width, the spin box is displayed with this width and spin buttons become

	larger (with greater width). If this value exceeds the default box width, it is ignored and spin buttons are displayed with their default widths (see **Tips & Tricks**).
Toggle start	Start position of “Toggle count” 0-termianted strings in string buffer. Used for “Toggle” type when “**Mode’** = “Label”.
Toggle count	Number of toggle states. Used for “Toggle” type only.
Toggle state	Initial toggle state. This value becomes current toggle state each time you exit the configuration dialog by OK. Used for “Toggle” type only.
Width	Width of spin tool in pixels (spin box + buttons).
Height	Height of spin box in pixels.
Depth	Depth of spin box in spin buttons (from 1 to 3).
Unsel bgnd color	Background color of unselected spin tool (not on focus): • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Unsel fgnd color	Foreground color of unselected spin tool (not on focus): • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Sel bgnd color	Background color of selected spin tool (on focus): • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Sel fgnd color	Foreground color of selected spin tool (on focus): • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Mode	GUI text mode (toggle text source) – string buffer or “Label” argument.
Label	Direct string argument for the “Toggle” tool type, used when “**Mode**” = label. Contains toggle texts in format “….~…..~…”. Ignored for integer and float spin.
Group id	Number of GUI group, the tool belongs to. This option is used in edit mode only. Place GUI tools, which overlap on the screen in different groups.

Results

Result	Description
Spin event	Spin event: • 0 : no spin event • 1 : spin tool put on focus • 2 : spin value or toggle state changed.
Spin value	Current spin value or toggle state.

Similar tools

Other GUI tools.

Usually combined with

Other GUI tools.

Load GUI tool texts from external file into the string buffer by the “Load string” tool.

Tips & Tricks

You can make larger spin buttons (needed to work with the touch-screen for example) by:

- Increase button height by setting greater “**Height**” argument.
- Increase button width by setting:
  “**Max edit length**” < (“**Width**” – 8 - 4 * ”**Depth**”) / 8.

Example:

The default spin width is 80 pixels and the default depth is 2 pixels. Use “**Max edit length**” value, which is less than 8 = (80 – 8 – 4*2) / 8 to make larger spin buttons.

Examples

Not indexed yet – sorry.

6.7. Radio-button menu

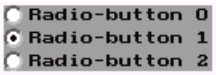
Group

GUI tools.

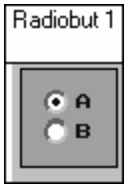
Short description

The tool displays a radio-button menu. Left mouse click checks (activates) on of the radio-button and resets all other.

VIMOS kernel presentation



Editor icons



Description

General function:

The tool displays a menu of 2 or more radio-buttons with explanation texts. One radio-button is always checked (set), and all other radio-buttons are not checked. You can set another radio-button by left mouse click on the button or on its text. This operation resets the previous checked radio-button. The radio-button texts are read from the string buffer or from the "**Label**" argument (see "*6.1.1. GUI texts*").

Results:

The tool returns 2 float results – radio-button event and radio-button value (index of checked radio-button).

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Point	Upper left corner of radio-button menu
Type	Type of radio-button menu: • **Vertical**. Vertical radio-button menu. • **Horizontal**. Horizontal radio-button menu.

Text start	Start position of “Count” 0-termianted radio-button strings in the string buffer, used when “**Mode**” = string buffer.
Count	Number of menu buttons (>= 2).
Value	Initial menu value (index of set button in the range [0,”Count”-1].
Bgnd color	Background radio-button color: • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Fgnd color	Foreground radio-button and text color: • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Mode	GUI text mode (radio-button text source) – string buffer or “Label” argument.
Label	Direct string argument for the radio-button texts, used when “**Mode**” = label. Contains texts in format “….~….~…”.
Group id	Number of GUI group, the tool belongs to. This option is used in edit mode only. Place GUI tools, which overlap on the screen in different groups.

Results

Result	Description
Radio-button event	Radio-button event: • 0 : no radio-button event • 1 : radio-button menu selected (put on focus) • 2 : radio-button value changed.
Radio-button value	Radio-button value – index of set radio-button: 0=first, 1=second, etc, up to (“Count” – 1).

Similar tools

Other GUI tools.

Usually combined with

Other GUI tools.

Load GUI tool texts from external file into the string buffer by the “Load string” tool.

Tips & Tricks

Use the check-box tool if you need 1-item menu.

Examples

Not indexed yet – sorry.

6.8. Check-box

Group

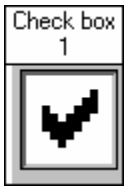
GUI tools.

Short description

The tool displays a check-box. Left mouse click toggles the check-box on/off.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool displays a check-box with an optional text. You can toggle the check-box state on/off by left mouse click on the check-box or on the text. The check-box text is read from the string buffer or from the "Label" argument (see "6*.1.1. GUI texts*").

Results:

The tool returns 2 float results – check-box event and check-box value (0 or 1).

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Point	Upper left corner of check-box.
Text start	Start position of check-box text in string buffer, used when "**Mode**" = string buffer.
Text length	Max length of check-box text when the text is not 0-terminated.
Value	Initial check-box value (0=off, 1=on).
Bgnd color	Background check-box color:

	• **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Fgnd color	Foreground check-box and text color: • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors.
Mode	GUI text mode (check-box text source) – string buffer or “Label” argument.
Label	Direct string argument for the check-box text, used when “**Mode**” = label.
Group id	Number of GUI group, the tool belongs to. This option is used in edit mode only. Place GUI tools, which overlap on the screen in different groups.

Results

Result	Description
Check-box event	**Check-box** event: • 0 : no check-box event • 1 : check-box selected (put on focus) • 2 : check-box value changed.
Check-box value	Check-box value: 0=off, 1=on

Similar tools

Other GUI tools.

Usually combined with

Other GUI tools.

Load GUI tool texts from external file into the string buffer by the “Load string” tool.

Tips & Tricks

Put several check-box tools if you want to create flag menu in which each flag can be set or not (the radio-button menu allows one flag to be set, all other are off).

Examples

Not indexed yet – sorry.

6.9. GUI rectangle

Group

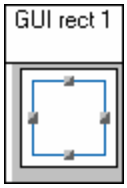
GUI tools.

Short description

The tool displays a rectangle, which can be moved, resized and rotated by the mouse in run mode.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool draws a rectangle. The rectangle can be moved, resized or rotated during user-program execution (in run mode). The tool supports three optional buttons, which can be dragged by the left mouse button to perform some operation:

- “Move” button, located on rectangle-position point (rectangle center, top-left corner or bottom-left corner). Drag this button to move the rectangle.
- “Resize” button, located on rectangle corner, which is opposite to the top-left corner (for center and top-left point) or the bottom-left corner (for bottom-left point). Drag this button to change rectangle dimensions (width and height).
- “Rotate” button, located outside the rectangle near the “Resize” button. Drag this button to rotate the rectangle. The drag operation continues while you keep pressed left mouse button, regardless of the cursor position (on the button or outside the button).

Each of the three buttons can be configured individually by three options:

- Off. The button is not shown and respective operation is disabled.
- On. The button is permanently shown and respective operation is enabled.
- Auto. The button is shown when mouse cursor is inside the rectangle or inside the button area. The respective drag operation is possible when the button is visible.

There are several tool types with different graphical representations on the screen:

- Rectangle border lines
- Rectangle filled with color
- Rectangle with middle vertical arrow (arrow up)

- Rectangle with middle horizontal arrow (arrow right)
- Marker - a cross, which connects the center points of the opposite rectangle sides
- Ellipse border (not supported yet)
- Ellipse filled with color (not supported yet).

Results :

The tool has several results (see result table below). The event result returns current cursor position in respect to the rectangle area, left/right clicks inside rectangle and states of the drag operations. The other results return current values of rectangle parameters, which may vary as a result of the user move/resize/rotate operations.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Point	Rectangle-position point.
Angle	Rotation angle (clockwise).
Type	Rectangle type: • **Rect**. Draw rectangle borders. • **Rect fill**. Fill rectangle with color. • **Rect v.arrow**. Draw rectangle borders and middle vertical arrow (up arrow). • **Rect h.arrow**. Draw rectangle borders and middle horizontal arrow (right arrow). • **Marker**. Draw cross, composed of two lines, which connect the middle points of the opposite rectangle sides. • **Ellipse**. Draw ellipse borders (not supported yet) • **Ellipse fill**. Fill ellipse with color (not supported yet).
Width	Rectangle width in pixels.
Height	Rectangle height in pixels.
Point pos	Point position: • **Center**. Point at rectangle center. • **Top left**. Point at top left rectangle corner. • **Bottom left**. Point at bottom left rectangle corner.
Move	Move button flag: • **Off**. Move button is disabled. Move operation is disabled. • **On**. Move button is permanently shown. Move operation is enabled. • **Auto**. Move button is shown when mouse cursor is inside rectangle or inside button area. Move operation is enabled when button becomes visible.
Resize	Resize button flag: • **Off**. Resize button is disabled. Resize operation is disabled. • **On**. Resize button is permanently shown. Resize operation

	is enabled. • **Auto**. Resize button is shown when mouse cursor is inside rectangle or inside button area. Resize operation is enabled when button becomes visible.
Rotate	Rotate button flag: • **Off**. Rotate button is disabled. Rotate operation is disabled. • **On**. Rotate button is permanently shown. Rotate operation is enabled. • **Auto**. Rotate button is shown when mouse cursor is inside rectangle or inside button area. Rotate operation is enabled when button becomes visible.
Color	Rectangle color: • **Default**. Use default color. • **Blue | Green | Cyan | . . .** Other user-selectable colors. • **None.** Don't draw rectangle, show buttons if enabled.
Group id	Number of GUI group, the tool belongs to. This option is used in edit mode only. Place GUI tools, which overlap on the screen in different groups.

Results

Result	Description
Event	Rectangle mouse event: • 0 : no rectangle event • 1 : left click inside rectangle • 2 : left double click inside rectangle • 3 : right click inside rectangle • 4 : mouse cursor enters rectangle • 5 : mouse cursor is inside rectangle • 6 : mouse cursor exits rectangle • 7 : rectangle is moved • 8 : rectangle is resized • 9 : rectangle is rotated
Point	Rectangle position point (center, top-left or bottom-left).
Angle	Rectangle rotation angle (clockwise)
Width	Rectangle width in pixels.
Height	Rectangle height in pixels.

Similar tools

Other GUI tools.

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

7. Statistical tools

There are different types of data storage within VIMOS. We try to keep their number down. However there exist different tasks within an image processing system and keeping track of the earlier results is one such task. Therefore there is a set of ‘Statistic counters’ or just counters within VIMOS. These counters do not loose their values at program start or end. They can be written to flash memory, refreshed from there and accessed not only from within the user program during ‘Run mode’ but also directly from the ‘Main’ menu within ‘Edit mode’.

The main purpose of these counters is to count events like ‘Good parts’, ‘Missing bolts’, ‘Total count’ and so on. Therefore there exist tools to increment and decrement a counter. The resulting actual count can be used in calculations and decisions within the user-program. So for instance production could be stopped if there were five broken bottles in a row.

Since the content of the counters ‘survives’ program halts and switches they are a good way to hand data over from one run to the other.

You may have expected to see statistical calculations or diagrams within this tool group. While both can be done with tools from other groups it is not so simple. We plan to add such tools to VIMOS as soon as possible - sorry.

Editor toolbar of this group of tools :

7.1. Reset all counters

Group

Statistical tools

Short description

The tool sets all counters back to zero.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons

Description

General function :

Resets all counters back to zero. This tool is not executed during 'Edit mode'.

Results :

This tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

This tool has no configuration dialog and no arguments.

Results

This tool has no results.

Similar tools

There are no similar tools.

Usually combined with

Tools from the ‘Statistic tools’ group.

Tips & Tricks

If you use counters at all within your program you should make sure that they are in a known state at the beginning of the execution. Since they are NOT reset automatically at program start better include the ‘Reset all counters’ tool at the beginning of your program.

Examples

Not indexed yet – sorry.

7.2. Set counter

Group

Statistical tools

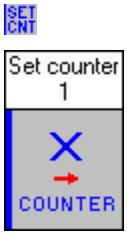
Short description

Set a counter to a given value.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

Sets a counter to a given value. This tool is not executed during 'Edit mode'.

Results :

This tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Counter	Which counter to set (number)
Value	Value, loaded into the selected counter (32 bit)

Note:

The dialog does not read counter values when opened. The 'Value' spin does not display the current value of the selected counter. Use 'Statistics' from the 'Main' menu to view/change counter values interactively.

Similar tools

Reset all counters	to set all counters to zero
Increment counter	to increase the current value of a counter by one
Decrement counter	to decrease the current value of a counter by one

Usually combined with

Tools from the ‘Statistic tools’ group.

Tips & Tricks

If you use counters at all within your program you should make sure that they are in a known state at the beginning of the execution. Since they are NOT reset automatically at program start better include the ‘Reset all counters’ tool at the beginning of your program.

Examples

Not indexed yet – sorry.

7.3. Increment counter

Group

Statistical tools

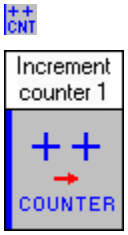
Short description

Increase the current value of a counter by one.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

Increment the actual value of a counter by one. This tool is not executed during 'Edit mode'.

Results :

This tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Counter	Which counter to increment (number)

Similar tools

Decrement counter	Reverse operation - decrease the current value of a counter by one
Reset all counters	to set all counters to zero
Set counter	to set a counter to a given value

Usually combined with

Tools from the ‘Statistic tools’ group.

Tips & Tricks

If you use counters at all within your program you should make sure that they are in a known state at the beginning of the execution. Since they are NOT reset automatically at program start better include the ‘Reset all counters’ tool at the beginning of your program.

Examples

Not indexed yet – sorry.

7.4. Decrement counter

Group

Statistical tools

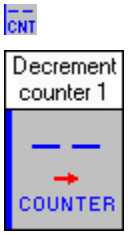
Short description

Increase the current value of a counter by one.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

Decrement the actual value of a counter by one. This tool is not executed during 'Edit mode'.

Results :

This tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Counter	Which counter to decrement (number)

Similar tools

Increment counter	Reverse operation - increase the current value of a counter by one
Reset all counters	to set all counters to zero
Set counter	to set a counter to a given value

Usually combined with

Tools from the ‘Statistic tools’ group.

Tips & Tricks

If you use counters at all within your program you should make sure that they are in a known state at the beginning of the execution. Since they are NOT reset automatically at program start better include the ‘Reset all counters’ tool at the beginning of your program.

Examples

Not indexed yet – sorry.

7.5. Save statistics

Group

Statistical tools

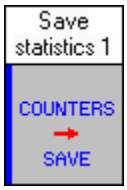
Short description

The tool writes the actual values of all the counters to flash memory file.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

Save the values of all counters into the statistics file **`st1.vm`** (default file name). This tool is not executed during 'Edit mode'. The 'Load' and 'Save' operations, executed in edit mode from the statistics menu, change the default file name.

Warning: This tool writes to file, which in case of flash memory could quickly consume all available space ! In that case you would have to leave VIMOS and use the 'Camera Files Functions' of the Simulator to delete some of the files (always include the last because it will be truncated) and then call a 'Pack Flash' operation to regain free flash memory. It's important to know that you will not be able to save your program and setup when leaving VIMOS if the flash is full. So better don't use this tool in 'Run-mode' without 'hands on' and by all means please save your work BEFORE you first start such a program. **New:** On the MMC (multimedia-card of the TI-based cameras) you can use this more freely. On the other cameras please use the new 'Delete Flash File' tool to pack the flash regularly after you called the tool described here.

Results :

This tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

This tool has no configuration dialog and no arguments.

Results

This tool has no results.

Similar tools

There are no similar tools.

Usually combined with

Tools from the ‘Statistic tools’ group.

Tips & Tricks

If you use counters at all within your program you should make sure that they are in a known state at the beginning of the execution. Since they are NOT reset automatically at program start better include the ‘Reset all counters’ tool at the beginning of your program.

Examples

Not indexed yet – sorry.

8. I/O Tools

The tools in this group are used for external communication with a PC or other devices like PLCs. These tools are hardware dependent. At present the cameras support serial communication via RS232/RS422, LAN communication via Ethernet and communication via the Beckhoff bus terminal system with digital inputs and outputs. The Ethernet option is possible on VC20xxE, VC40xxE and FA45 cameras. The PC Simulator can use the serial port or the LAN to communicate with the cameras. Since the PC has no PLC hardware, the Simulator simulates the PLC inputs and outputs of the camera by green and red LEDs. You can change the PLC inputs by clicking on the green LEDs (the left group of 4 LEDs).You can see the simulated status of the PLC outputs by the red LEDs (the right group of 4 LEDs).

The Beckhoff bus terminal system can be connected to the serial port of the camera or the PC. The Simulator will perform real I/O transfer by the get/set I/O box tools.

***ATTENTION**. All I/O tools except the "Clean receive Buffer" tool work in **run-mode** only.*

Editor toolbar of this group of tools :

8.1. I/O devices

The general-purpose I/O tools (all tools except PLC and Beckhoff ones) have a device argument, which specifies a source or destination I/O device. Depending on the device type the tools work in two modes:

- Serial mode – when using serial (RS-232) I/O device – possible on PC and cameras with serial port.
- TCP mode – when using TCP/IP I/O device – possible on PC and Ethernet camera VC20xxE.

Next sections in this chapter describe how to configure and use serial and TCP devices.

8.1.1. Serial I/O devices

By default, when you start the user program, all devices are serial. No matter what is the device number, an I/O tool transfers data to/from the serial port of the camera or the selected COM port of the PC (see "8*.1.1.2. Enabling execution of serial I/O tools in Simulator*"). The TCP configuration tools change the device type from serial to TCP dynamically (see "8*.1.2. TCP I/O devices*"). Note that when a TCP device is disconnected, it becomes a serial device.

NOTE: On TI-based cameras with Ethernet port like VC20xxE and VC40xxE the default serial port is actually the Telnet TCP port 23, used to communicate with the camera shell. So if you send a string to the default serial port, you can see the string printed on the terminal. This port is used to control VIMOS by mouse commands from the Simulator.

Some TI-based cameras like VC40xxE and FA45 have both Ethernet and RS232/RS422 ports. The "Open Serial Device" tool opens a serial device for I/O transfer via this RS232/RS422 port. Remember that this serial device is **NOT** the default serial device, which on Ethernet cameras is always the TCP port 23.

8.1.1.1. Enabling execution of serial I/O tools on camera

To enable execution of I/O tools in serial mode on camera, you should configure the user-program for an external I/O device. In the Editor - enter **File > Properties > Properties tab**, select other I/O device and baud rate. In VIMOS kernel - enter the "Configure Serial Device" dialog from **Edit main menu > Configuration > Serial device**. Set the following options:

- I/O device = Other I/O device
- I/O baud rate = desired baud rate, used by the serial port (from 1200 to 115200).

After setting these options you should save the user-program and restart VIMOS.

***ATTENTION**. Remember that starting VIMOS on serial camera with such settings disables the mouse commands. To regain serial control over VIMOS, restart the system with connected/simulated mouse (PC), so that VIMOS can recognize a mouse during its initialization stage. You need to move the mouse a bit during VIMOS startup so that it generates some messages:*

- *On ADSP camera - move the mouse when the system logo appears on the camera monitor.*
- *On TI camera – move the mouse when the message "Searching mouse…" is displayed on the camera monitor*

No need to move it much. Make a 'right click' to exit run-mode when the logo disappears. Since at present the exiting from run-mode needs up to three full cycles you may need to generate trigger signals to get the menu. The same may have to be repeated again after you did a 'left click' on 'Exit run mode'. While doing all this you may move the mouse just a bit. Don't move it too much because then the clicks you did will be lost. Sorry we'll have to fix this, but after doing it a few times you'll have no

more trouble.

Setting the “Other I/O device” option is not necessary for serial devices, opened by the “Open Serial Device” tool. Working with such devices is possible on VC40xxE and FA45 cameras, which have Ethernet and RS2323/RS422 port.

8.1.1.2. Enabling execution of serial I/O tools in Simulator

By default execution of I/O tools is disabled in Simulator. To enable execution of serial I/O tools, enter **Camera > Simulation Options** and select **Serial port** = COM1 to COM8. All serial I/O devices (the device number is ignored) are directed to the selected COM port.

8.1.2. TCP I/O devices

By default all I/O devices are serial. The TCP configuration tools described below change a device type from serial to TCP (see “8*.1.2.3. Configuring TCP devices*”).

8.1.2.1. Enabling execution of TCP I/O tools on camera

Execution of TCP I/O tools is possible on Ethernet camera VC20xxE. The mouse control is not disabled since TCP tools and mouse use different TCP ports (mouse commands are passed through the telnet port 23). It is not necessary to set other I/O device and baud rate as required for the serial tools.

8.1.2.2. Enabling execution of TCP I/O tools in Simulator

By default execution of I/O tools is disabled in Simulator. To enable execution of I/O tools with TCP device, enter **Camera > Simulation Options** and select **Ethernet port** = TCP.

8.1.2.3. Configuring TCP devices

The following tools configure dynamically TCP devices during the execution of the user program:

- **Create Server Device.** The tool does not fulfill any connections, it just setups an I/O device as a TCP server. Server devices accept connection requests from remote clients automatically in run mode. A connection request is accepted if the remote peer has IP address, equal to the IP address specified by the tool. If the selected device has been already assigned as a TCP server or client, the tool ignores the previous settings, disconnects the device and configures it as a TCP server device.
- **Client Device Connect.** The tool configures a device as client and connects to a remote server. If the selected device has been already configured as TCP server or client, the function ignores the previous settings, disconnects the device and connects it as TCP client to a server with specified IP address and port.

***ATTENTION**. Stopping of run mode and entering into edit mode disconnects all TCP devices and closes the created sockets. When a TCP device is disconnected, it becomes a default serial device.*

Currently you can use one fixed connection port – port 2000.

***TIPS & TRICKS**. To disconnect a connected server or client device simply reconfigure it as server/client with non-existing IP address.*

8.2. Get I/O value

Group

I/O tools.

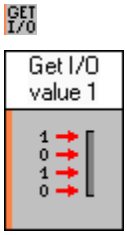
Short description

This tool reads the 4 input PLC lines.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool reads the 4 input PLC lines. It then generates a result where the user can specify what value a given bit of the result should take. The arguments 'Bit 0', 'Bit 1', 'Bit 2', 'Bit 3' specify how bits 0,1,2,3 of the result are generated. Bit 0 is the least significant (LSB) and Bit 3 the most significant (MSB). Any of these bits may get assigned a constant value ('#0' or '#1') or the actual state of an PLC input line (0,1,2,3).

***ATTENTION**. Some cameras like sensors M40/M50 or TI cameras without video-output like FA45/VC40xx have 2 input PLC lines IN0 and IN1. The tool returns 2-bit result for these cameras in the range 0-3.*

Results :

The result is a number in the range 0 – 15, equal to an integer, composed of the four bit values.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Bit 0	Specifies how bit 0 of the result is generated
Bit 1	Specifies how bit 1 of the result is generated

Bit 2	Specifies how bit 2 of the result is generated
Bit 3	Specifies how bit 3 of the result is generated

Results

Result	Description
PLC_IN	Up to four active bits show the status of the PLC inputs

Similar tools

Get I/O box value same operation but using an external I/O box with 8 PLC inputs

Set I/O values reverse operation – sets the PLC outputs

Usually combined with

IF-constructions to make decisions, depending on the state of the PLC inputs.

Tips & Tricks

If you find the dialog confusing and the explanation here not much better, then:

- in VIMOS kernel – set the four arguments to 0, 1, 2 and 3
- in Editor - set the four arguments to 2, 3, 4 and 5 to select PLC lines 0 to 3

and the result will be a number where every bit represents the corresponding PLC input.

Examples

Not indexed yet – sorry. (Most examples use PLC inputs for triggering.)

8.3. Set I/O value

Group

I/O tools

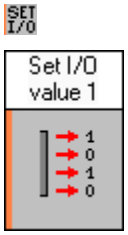
Short description

This tool sets the 4 output PLC lines.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The command sets the 4 output PLC lines. The user can specify the value for every line. Allowed values are '#0' (reset to zero), '#1' (set to one) and 'Unchanged' (the state of the output line remains unchanged).

Results :

This tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Out 0	Specifies the value of output line 0
Out 1	Specifies the value of output line 1
Out 2	Specifies the value of output line 2
Out 3	Specifies the value of output line 3

Results

This tool has no results.

Similar tools

Set I/O box value	same operation but using an external I/O box with 8 PLC outputs
Get I/O values	reverse operation – reads the PLC inputs

Usually combined with

The tool is normally used in branches of IF-constructions to represent certain states of the program.

Tips & Tricks

There is nothing special to say about this tool.

Examples

Not indexed yet – sorry.

8.4. Get I/O Box Value

Group

I/O tools

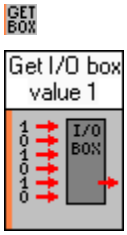
Short description

This tool reads up to 8 digital inputs of an external I/O box (module).

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool works with the Beckhoff bus terminal system. Visit www.beckhoff.com to learn more about Beckhoff bus terminals (refer to the document "Bus coupler with RS232 interface BK8100"). The bus terminal system consists of an intelligent "bus coupler" device BK8100, which is connected to the serial port, and several input/output bus terminals (called further I/O modules) connected to the bus coupler. The modules hold input/output channels. The system is highly scalable – you can connect arbitrary number of I/O modules (in a given reasonable range) to the bus coupler to fit your practical needs. There are four basic module types:

- Digital input modules
- Digital output modules
- Analog input modules
- Analog output modules

Each module holds several channels – 2, 4 or 8 for example. Analog channels are 16-bits wide. Digital channels (PLC lines) are 1 bit wide. You may install both input and output modules and both analog and digital modules in one bus terminal system. The modules are numbered starting from 1 (POS01) according to their physical position on the mounting rack. The last module on the rack should be an end terminal, which has no I/O channels.

***ATTENTION**. Currently VIMOS works with digital I/O modules only with up to 8 channels each. Analog modules are not supported but may be present in the bus system. All installed I/O modules should be described in the configuration file **iobox.vm** (see description below).*

*By default the BK8100 bus coupler is configured with "even parity" and baud rate 38400. You should use a bus coupler, which is configured with "**no parity**". Remember to configure VIMOS with the respective baud rate.*

VIMOS reads the description of the bus terminal system from a text configuration file with the name ***iobox.vm***. Below is given the format of the file. The module description lines like:

```
digital_in  = . . .
digital_out = . . .
. . . . . . . . . . .
```

should be ordered according to the actual mounting positions of the modules POS01, POS02, ... These positions should be used to access a given module by the "**Module**" argument.

```
;-------------------------------------------------------------------------------
; IOBOX.VM = Beckhoff I/O module configuration file.
; Format:
;   bus_ad = xx            ; bus coupler address, xx = 1-99
;   time_out = xxxx        ; response time out in ms
;
;   analog_out  = xx       ; analog output module, xx = number of channels
;   analog_in   = xx       ; analog input module, xx = number of channels
;   digital_out = xx       ; digital output module, xx = number of channels
;   digital_in  = xx       ; digital input module, xx = number of channels
;
; Module description lines in this configuration file should be ordered from top
; to bottom according to the physical mounting positions POSxx, xx=01,02,... of
; the modules. Visit www.beckhoff.com to learn more about Beckhoff bus terminals
; (refer to the document "Bus coupler with RS232 interface BK8100").
;
; Number of bits per channel:
;      Analog channel = 16 bits
;      Digital channel = 1 bit
;
; Note: Both lowercase and uppercase keywords are accepted.
;
; Example configuration file is shown below:
;-------------------------------------------------------------------------------
bus_ad = 1          ; POS00 : bus coupler address [1-99]
time_out = 600      ; response time out in ms

analog_out  = 2     ; POS01 : analog output, 2 channels (16-bit word each)
analog_in   = 1     ; POS02 : analog input, 1 channel
digital_out = 8     ; POS03 : digital output, 8 channels (1 bit each)
digital_in  = 6     ; POS04 : digital input, 6 channels
analog_out  = 1     ; POS05 : analog output, 1 channel
analog_in   = 2     ; POS06 : analog in, 2 channels
digital_out = 4     ; POS07 : digital output, 4 channels
digital_in  = 4     ; POS08 : digital input, 4 channels
```

Create a text file **iobox.vm** in the format described above, which describes the actual hardware configuration of the bus terminal system. Copy the file into the Simulator workspace folder for work on PC with Simulator. Upload the file into camera flash for work on camera by the "Send" button of the "Camera Files Functions" dialog of the Simulator in mode "Binary data". Remember to copy this file to MD drive if you change the default file drive of TI camera from FD to MD.

***ATTENTION**. When VIMOS is started in mode "Other I/O device" and the file **iobox.vm** is found in the flash, then VIMOS performs initialization of the bus system by setting all output modules to known state 0. If you are working with external I/O device other than the bus system, you should delete **iobox.vm** from flash.*

The tool reads up to 8 channels (PLC lines) of the specified digital input module. It generates a result where each bit holds the value of one input channel (0 or 1). The first 8 arguments "**Bit 0**" to "**Bit 7**" specify how respective bits in the result "**Input_value**" are formed:

- **#0** - set result bit to 0
- **#1** - set result bit to 1
- **chan1** - set result bit from channel 1
- **chan2** - set result bit from channel 2
- . . . . .

- **chan8** - set result bit from channel 8

Bit 0 is the least significant bit (LSB) of the result and Bit 7 the most significant bit (MSB).

***ATTENTION**. If the module has less than 8 channels (4 for example), you should set result bits from the available channels 1 to 4 only. Values of missing channels are set to 0. The "**Chan_count**" result returns the actual number of module channels.*

The "**Module**" argument specifies the module, which is used to read channel data: 1 for module at position POS01, 2 for POS02 and so on. Select a number of ***input*** digital module. In case of invalid or missing module number the tool results with error **2110**.

The "**Update**" argument specifies how to read input channel data. The "**No**" option reads module data from internal VIMOS memory buffer without doing an I/O operation. The "**Yes**" option does I/O operation – it reads the data of all input modules in the bus system into internal VIMOS buffer and then reads the data of the specified module into the tool result.

***INFORMATION**. This method for reading module data is caused by the peculiarities of the data communication with the Beckhoff bus terminal system. One I/O operation writes all output module data and reads all input module data.*

If the bus system contains several digital input modules, you must execute one tool for each one of the input modules. The recommended best-speed execution sequence is:

- Read data of a given module with option "**Update**" = "**Yes**". This operation will update the data of all input modules in the internal VIMOS memory buffer.
- Read data of all remaining modules with option "**Update**" = "**No**".

Note: The bus system is connected to camera or PC via the serial interface (RS232). Use the following settings to configure VIMOS to work with an external I/O device (see chapter introduction for details):

- I/O device = Other I/O device
- I/O baud rate = 38400

Please check the availability of that special hardware before becoming dependent on it. This tool works on TI camera and PC Simulator.

Results :

The tool returns 3 float results:

- **Input_value** – number in the range [0,255], equal to an integer, composed of the 8 bit values.
- **Chan_count** – number of channels of selected module.
- **Status** – I/O status, returned by the last I/O operation in mode "**Update**" = "**Yes**". Refer to the document "Bus coupler with RS232 interface BK8100" for more details.

Except an error status, the tool may return a number of result error codes – see the "Errors" table below.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Bit 0	Specifies how bit 0 of the result is set.
Bit 1	Specifies how bit 1 of the result is set.

Bit 2	Specifies how bit 2 of the result is set.
Bit 3	Specifies how bit 3 of the result is set.
Bit 4	Specifies how bit 4 of the result is set.
Bit 5	Specifies how bit 5 of the result is set.
Bit 6	Specifies how bit 6 of the result is set.
Bit 7	Specifies how bit 7 of the result is set.
Module	Number of digital input module (position on the rack).
Update	Specifies reading mode – without or with an I/O operation, which re-reads data of all input modules.

Results

Result	Description
Input_value	Combined bits of input digital channels
Chan_count	Number of module channels.
Status	I/O status (updated on each execution with option Update=Yes)

Errors

Error	Macro	Description
2100	IOB_R_NO_MEMORY	Memory allocation error
2101	IOB_R_NO_CONFIG	Configuration file not found
2102	IOB_R_INV_FORMAT	Invalid format of configuration file
2103	IOB_R_DATSIZ_OVF	Total data size overflow (>255)
2104	IOB_R_MODBUF_OVF	Module buffer overflow – number of configuration file modules > IOB_MODULE_CNT
2105	IOB_R_DATABUF_OVF	Out/in data buffer overflow
2106	IOB_R_INTERNAL_ERR	Internal error
2107	IOB_R_TIME_OUT	Response time out
2108	IOB_R_CFG_MISMATCH	Mismatch b/n configuration file and received data size
2109	IOB_R_INV_CHECKSUM	Response checksum mismatch
2110	IOB_R_INV_MODNUM	Invalid module number for data access
2111	IOB_R_MODDATA_OVF	Module data buffer overflow

Similar tools

Get I/O values same operation but using the four PLC inputs of the camera

Set I/O box value reverse operation – writes up to 8 channels of a digital output module

Usually combined with

IF-constructions to make decisions, depending on the state of the digital inputs.

Tips & Tricks

If you find the dialog confusing and the explanation here not much better, just use the default arguments values of “**Bit 0**” to “**Bit 7**” in Editor or set these arguments to “chan1”,…,chan8 in VIMOS kernel. The result “**Input_value**” will be a number where every bit is set from the corresponding input channel.

Examples

Not indexed yet – sorry.

8.5. Set I/O Box Value

Group

I/O tools

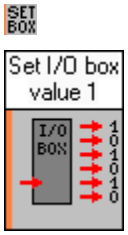
Short description

This tool writes up to 8 digital outputs of an external I/O box (module).

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool works with the Beckhoff bus terminal system. Please read the description of the "**Get I/O Box Value**" tool to learn more about the bus terminal system and how it is configured in VIMOS.

The tool writes up to 8 channels (PLC lines) of the specified digital output module. It generates a result where each bit holds the actual value written to one output channel (0 or 1). The first 8 arguments "**Out 0**" to "**Out 7**" specify values, which are written to channels 1 to 8 and how bits 0-7 in the result "**Output_value**" are formed:

- **#0** - set respective channel (and result bit) to 0
- **#1** - set respective channel (and result bit) to 1
- **Unchanged** – leave respective channel unchanged (respective result bit is set from current channel value)

"**Out 0**" sets channel 1, "**Out 1**" sets channel 2 and so on. "**Out 0**" sets the least significant bit (LSB) of the result and "**Out 7**" sets the most significant bit (MSB).

***ATTENTION**. If the module has less than 8 output channels (4 for example), you should set the available channels 1 to 4 only. Values of missing channels are ignored. The "**Chan_count**" result returns the actual number of module channels.*

The "**Module**" argument specifies module number: 1 for module at position POS01, 2 for POS02 and so on. Select a number of an ***output*** digital module. In case of invalid or missing module number the tool results with error **2110**.

The "**Update**" argument specifies how to write output channel data. The "**No**" option writes module data into internal VIMOS memory buffer without any I/O operation. The "**Yes**" option does I/O operation – it first writes the data of the specified module into internal VIMOS buffer and then sends all

output data from the memory buffer to the output modules. On VIMOS start all output channels are set to known state 0.

***INFORMATION**. This method for writing module data is caused by the peculiarities of the data communication with the Beckhoff bus terminal system. One I/O operation writes all output module data and reads all input module data.*

If the bus terminal system contains several digital output modules, you must execute one tool for each one of the output modules. The recommended best-speed execution sequence is:

- Write data to all output modules except one (the last) with option "**Update**" = "**No**".
- Write data to last module with option "**Update**" = "**Yes**". This operation will write the data from the internal VIMOS memory buffer to all output modules.

Note: The bus system is connected to camera or PC via the serial interface (RS232). Use the following settings to configure VIMOS to work with an external I/O device (see chapter introduction for details):

- I/O device = Other I/O device
- I/O baud rate = 38400

Please check the availability of that special hardware before becoming dependent on it. This tool works on TI camera and PC Simulator.

Results :

The tool returns 3 float results:

- **Output_value** – number in the range [0,255], equal to the integer, composed of the 8 output bits (the actual value written to the output module).
- **Chan_count** – number of channels of selected module.
- **Status** – I/O status, returned by the last I/O operation in mode "**Update**" = "**Yes**". Refer to the document "Bus coupler with RS232 interface BK8100" for more details.

Except an error status, the tool may return a number of result error codes – see the "Errors" table in the description of the "Get I/O Box Value" tool.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Out 0	Value of channel 1 (bit 0 of result).
Out 1	Value of channel 2 (bit 1 of result).
Out 2	Value of channel 3 (bit 2 of result).
Out 3	Value of channel 4 (bit 3 of result).
Out 4	Value of channel 5 (bit 4 of result).
Out 5	Value of channel 6 (bit 5 of result).
Out 6	Value of channel 7 (bit 6 of result).
Out 7	Value of channel 8 (bit 7 of result).
Module	Number of digital output module (position on the rack).
Update	Specifies writing mode – without or with an I/O operation, which writes data to all output modules.

Results

Result	Description
Output_value	Actual value written to the output module (combined bits).
Chan_count	Number of module channels.
Status	I/O status (updated on each execution with option Update=Yes)

Errors

See the “Errors” table in the description of the “**Get I/O Box Value**” tool.

Similar tools

Set I/O values same operation but using the four PLC outputs of the camera

Get I/O box value reverse operation – reads up to 8 channels of a digital input module

Usually combined with

The tool is normally used in branches of IF-constructions to represent certain states of the program.

Tips & Tricks

There is nothing special to say about this tool.

Examples

Not indexed yet – sorry.

8.6. Clean Receive Buffer

Group

I/O tools

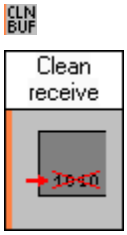
Short description

Deletes all data in the receive buffer of the specified I/O device.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool cleans the receive buffer (all available data received so far) of the specified I/O device. The tool works both in run and edit mode.

Depending on the device configuration, the tool works in two modes – serial or TCP.

***ATTENTION**. Working with serial and TCP I/O devices requires special setup. Please read I/O device description in "*8.1. I/O devices*".*

.

Serial mode:

The serial interface works in parallel with the other hardware of the camera. It is able to receive while the processor is busy with other tasks. During such a period the received data is written into a receive buffer that holds 63 bytes on ADSP camera and 256 bytes on TI camera. If greater number of bytes is received before the system can process that input data, the buffer will overflow and data will be lost. The problem here is, that additionally to the loss of some of the data, the data remaining in the buffer will be out of synchronization and most probably useless.

You can create such a situation by executing a VIMOS program that needs a few seconds for each cycle and then at the beginning of the cycle click the right mouse button and then move the mouse a lot. The mouse movement will generate a lot of new serial data and since VIMOS is still not finished with the actual cycle the right click at the beginning will be lost from the input buffer and the menu will not appear. Since the data in the buffer will be invalid now you'll have to wait a few cycles before the buffer will be empty again and you can do a successful click.

The data in the serial input buffer will remain there until either it is fetched (processed) or the power of the camera is turned off. So it is possible that at the start of your program there is useless data within

that buffer and this would harm your processing of serial data. So it's a good idea to clean the serial receive buffer during the first cycle if you use serial receive tools within your program.

TCP mode:

The tool reads all available bytes from the specified TCP device (server or client), without any waiting for a new data to come.

Results :

This tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Device	I/O device number in the range [0,255].

Results

This tool has no results.

Similar tools

There are no similar tools.

Usually combined with

I/O "receive" tools when you want to reset the interface.

Tips & Tricks

There's nothing special to say about this tool.

Examples

Not indexed yet – sorry.

8.7. Receive Result

Group

I/O tools

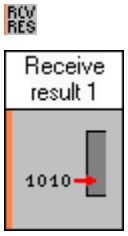
Short description

This tool receives a value from the specified I/O device.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool receives a value from the specified I/O device. Depending on the type of the I/O device, the tool works in two modes – serial or TCP.

***ATTENTION**. Working with serial and TCP I/O devices requires special setup. Please read I/O device description in "*8.1. I/O devices*".*

.

The sender of the value should be a "Send result" tool working on another camera or PC. In serial mode the two tools must have equal baud rates.

TCP mode:

Some tool arguments are ignored in TCP mode. The "**Wait**" argument specifies how to wait for incoming data:

- **Don't wait** – don't wait if incoming data is not present
- **Wait in mS** – wait "Wait in mS" milliseconds for incoming data
- **Wait forever** - wait forever for incoming data

The type of the expected value is received from the sender. The tool fills respective result depending on the received type.

Results :

The tool returns the received data depending on the selected type.

WARNING: There is a general rule, which is observed in the generation of all VIMOS tool results. When a given tool returns an error result, a non-zero error code is associated with the result. Although the tool writes some new result value, other tools which have linked arguments to this error result, will use the last valid tool result, produced by the tool without error, and not the newly saved value.

All I/O "***receive***" tools return errors (timeout or others) when there is no incoming data from the remote I/O device. **Don't rely on** result values (for example: number of received bytes = 0) to check if the tool has received or not some data. **Always check tool errors first and then check data counts or other result values !**

Algorithm (serial mode)

The data packet includes the follow fields :

<Start byte>, <Type of data >, <Data>, <Checksum>

You can specify the value of the 'Start byte' within the dialog. The 'Type of data' field is 2 bytes long and can be:

- 00 - Result R (float) – data field is 4 bytes in BIN protocol, 8 – bytes in HEX;
- 01 - Point P (4 x short) – data field is 8 bytes in BIN protocol, 16 – bytes in HEX;
- 02 - Angle A (short) – data field is 2 bytes in BIN protocol, 4 – bytes in HEX;

The 'Checksum' is the least significant byte of the sum of all bytes from the data field. The bytes in the data field are ordered according to the "big endian" convention (MSB first).

The ASCII data types (see "APPENDIX A. ASCII table") specify special diagnostic mode of operation for work with terminal:

- <Start byte> is processed only if "Wait" specifies "Wait forever".
- The tool waits to receive an ASCII string, terminated by a carriage return (CR = 0x0D). Each received character is echoed to the interface. The Backspace key (code 0x08) could be used to discard last entered byte. In "ASCII prompt" mode the tool sends a prompt string (e.g. "R=": for float value type) before receiving the CR-terminated string.
- Checksum is not implemented.

Arguments

Argument	Description
Value type	Select type of value to receive : Float (R), Point (P), Angle (A). **Ignored in TCP mode.**
Start byte	Select start byte from 0 to 255. The tool checks for a start byte and if it is present, starts receiving the next bytes. **Ignored in TCP mode.**
Data type	• HEX – each byte is encoded by two hex symbols [0,1, ..9,A,B..F] • BIN – bytes are transferred directly • ASCII – the result value is received as CR-terminated ASCII string (see "APPENDIX A. ASCII table"). • ASCII prompt – same as ASCII but preceded by a prompt string. **Ignored in TCP mode.**
Checksum	Select checksum - Yes or No (one byte in BINARY format or two bytes in ASCII format). **Ignored in TCP mode.**
Wait	Specifies wait mode for acknowledgement of I/O data:

	• **Don't wait** – don't wait for acknowledgement • **Wait in mS** – wait "Wait in mS" milliseconds for acknowledgement • **Wait forever** – wait forever for acknowledgement
Wait in mS	Wait time in milliseconds used when selected "Wait" option is "Wait in mS".
Device	I/O device number in the range [0,255].

Results

Result	Description
Result	If the type of the received value is 'Result', the received data is available through this tool result.
Point	If the type of the received value is 'Point', the received data is available through this tool result.
Angle	If the type of the received value is 'Angle', the received data is available through this tool result.

Similar tools

Receive point-list	receive a point-list via I/O device
Send result	reverse operation – sends values via I/O device
Send point-list	send a part of the point-list via I/O device
Send image area	send a part of the image (Frame buffer) via I/O device

Usually combined with

All tools that accept data of the different types.

Clean receive buffer	deletes all data from the receive buffer of the I/O device

Tips & Tricks

If it doesn't seem to work in serial mode, please check other baud-rates (default 9600). Additionally please make sure you did successfully reconfigure the serial interface as described in "*8.1. I/O devices*".

If you use ASCII transfer and no checksum there will be a lot of codes that you can use for the 'Start byte' and that will not be present in the 'Data type' field or in the 'Data' field. 'Data type' only uses 0x00 ... 0x03 and in case of ASCII transfer 'Data' only contains 0x30 ... 0x39 and 0x41 ... 0x46 (see "APPENDIX A. ASCII table"). All the other codes are usable for unique 'Start bytes'. Since the 'Receive result' tool starts receiving only after receiving the right 'Start byte' you can send in a 'multi drop' situation – many cameras on one bus. Please note that 0x00 ... 0xFF are representations of one byte hexadecimal numbers equaling 0 ... 255 in decimal representation. To create a bus you need RS232 to RS485 converters.

Examples

Not indexed yet – sorry.

8.8. Send Result

Group

I/O tools

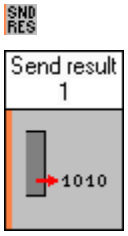
Short description

This tool sends a value to the specified I/O device.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool sends a value to the specified I/O device. Depending on the type of the I/O device, the tool works in two modes – serial or TCP.

***ATTENTION**. Working with serial and TCP I/O devices requires special setup. Please read I/O device description in "*8.1. I/O devices*".*

.

The receiver of the value should be a "Receive result" tool working on another camera or PC. In serial mode the two tools must have equal baud rates.

TCP mode:

Some tool arguments are ignored in TCP mode.

Results :

The tool returns a value that is zero for successful transfer and otherwise gives an error code.

Algorithm (serial mode)

The data packet includes the following fields :

<Start byte>, <Type of data >, <Data>, <Checksum>

You can specify the value of the 'Start byte' within the dialog. The 'Type of data' field for types HEX and BIN is 2 bytes long and can be:

- 00 - Result R (float) – data field is 4 bytes in BIN protocol, 8 – bytes in HEX;
- 01 - Point P (4 x short) – data field is 8 bytes in BIN protocol, 16 – bytes in HEX;
- 02 - Angle A (short) – data field is 2 bytes in BIN protocol, 4 – bytes in HEX;
- 03 - Counter C (long) – data field is 4 bytes in BIN protocol, 8 – bytes in HEX;

When the 'Send data' option is 'No' the tool will not send the 'Type of data' and 'Data' fields.

When the 'Checksum' option is 'No' the tool will not send the 'Checksum' field. In that case the 'Start byte' can be used as the information field. So you can generate 'free style' binary sequences. On the other hand if the 'Send start byte' option is 'No' the tool will not send the 'Start byte' field.

The 'Checksum' is the least significant byte of the sum of all bytes from the data field. The bytes in the data field are ordered according to the "big endian" convention (MSB first).

The ASCII data types (see "APPENDIX A. ASCII table") specify special diagnostic mode of operation for work with terminal:

- <Start byte> filed is sent if enabled
- <Type of data > field is not sent
- The value is converted to an ASCII string, which is sent vi the serial interface. In ASCII prompt mode the tool first sends a prompt string (for example "R="), followed by the value string.
- If "Checksum" is on, the sends space after the ASCII string.

Format of ASCII strings:

- For float result type: `xxxx.ddd`
- For point result type: `xxxx.ddd yyyy.ddd`
  where: `xxxx` and `yyyy` are x and y coordinates,
  `ddd` are x and y sub-pixel coordinates
- For angle type: `xxxx`
- For counter type: `xxxxxxxx`

Arguments

Argument	Description
Value	Data to be sent – usually must be linked to a tool result.
Start byte	Select start byte from 0 to 255. **Ignored in TCP mode.**
Send start byte	Yes or No. **Ignored in TCP mode.**
Data type	• HEX – each byte is encoded by two hex symbols [0,1, ..9,A,B..F] • BIN – bytes are transferred directly • ASCII – data is first converted to an ASCII string and the string is sent via the serial interface (see "APPENDIX A. ASCII table") • ASCII prompt – same as ASCII but preceded by a prompt string. **Ignored in TCP mode.**
Send data	Yes or No. **Ignored in TCP mode.**
Checksum	Select checksum Yes / No (one byte in BINARY format or two bytes in ASCII format).

	Ignored in TCP mode.
Device	I/O device number in the range [0,255].

Results

Result	Description
Success	Return result of sending – OK or Error

Similar tools

Send point-list	send a part of the point-list via I/O device
Send image area	send a part of the image (Frame buffer) via I/O device
Receive result	reverse operation – receives values via I/O device
Receive point-list	receive part of the point-list via I/O device

Usually combined with

All tools that produce results of different types.

Tips & Tricks

If it doesn't seem to work in serial mode, please check other baud-rates (default 9600). You may use a terminal program for that purpose. If you use ASCII mode all the transferred data will be readable with the exception of the checksum. You could set the 'Start byte' to 0x0A (10) or 0x0D (13) to get a new line in the terminal for every new message. While readable the data will not be easily decodable without some thinking. Please have a look at the data formats as described in the 'Description' chapter above.

Additionally please make sure you did successfully reconfigure the serial interface as described in "*8.1. I/O devices*".

If you use ASCII transfer and no checksum there will be a lot of codes that you can use for the 'Start byte' and that will not be present in the 'Data type' field or in the 'Data' field. 'Data type' only uses 0x00 ... 0x03 and in case of ASCII transfer 'Data' only contains 0x30 ... 0x39 and 0x41 ... 0x46 (see "APPENDIX A. ASCII table"). All the other codes are usable for unique 'Start bytes'. Since the 'Receive result' tool starts receiving only after receiving the right 'Start byte' you can send in a 'multi drop' situation – many cameras on one bus. Please note that 0x00 ... 0xFF are representations of one byte hexadecimal numbers equaling 0 ... 255 in decimal representation. To create a bus you need RS232 to RS485 converters.

Examples

Not indexed yet – sorry.

8.9. Receive String

Group

I/O tools.

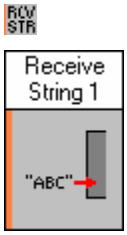
Short description

This tool receives a string from the specified I/O device.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool receives an ASCII string from the specified I/O device and stores the string into the string buffer (see "APPENDIX A. ASCII table"). Depending on the type of the I/O device, the tool works in two modes – serial or TCP.

***ATTENTION**. Working with serial and TCP I/O devices requires special setup. Please read I/O device description in "*8.1. I/O devices*".*

In serial mode the string bytes are received without any handshake (terminated by CR=0x0D). The sender of the string could be a "Send String" tool, working on another camera or a PC. The two tools must have synchronized transfer rates. The string terminating byte CR (0x0D) is not stored into the string buffer.

***ATTENTION**. The main purpose of the tool in string mode is diagnostic – to provide string input from a terminal. For this purpose it echoes received characters back to the sender. If you are using the tool in normal serial I/O between PC and camera, the sender should discard echoed characters (by the "Clean receive buffer" tool for example).*

In TCP mode the tool does not echo received characters because VIMOS uses a dedicated TCP port different from terminal port 23.

Results :

The tool returns the number of received string bytes excluding the terminating byte.

WARNING: There is a general rule, which is observed in the generation of all VIMOS tool results. When a given tool returns an error result, a non-zero error code is associated with the result. Although

the tool writes some new result value, other tools which have linked arguments to this error result, will use the last valid tool result, produced by the tool without error, and not the newly saved value.

All I/O "***receive***" tools return errors (timeout or others) when there is no incoming data from the remote I/O device. **Don't rely on** result values (for example: number of received bytes = 0) to check if the tool has received or not some data. **Always check tool errors first and then check data counts or other result values !**

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Start	Start index in the string buffer where to put the received string.
Wait	Wait time in ms (0=no wait).
Device	I/O device number in the range [0,255].

Results

Result	Description
String length	Length of received string without the terminating byte.

Similar tools

Receive result receives values via I/O device

Usually combined with

"Send String" tool, running on another camera or PC that communicates with your program.

Tips & Tricks

If it doesn't seem to work in serial mode, please check other baud-rates (default 9600). Additionally please make sure you did successfully reconfigure the serial interface as described in "*8.1. I/O devices*".

Examples

Not indexed yet – sorry.

8.10. Send String

Group

I/O tools.

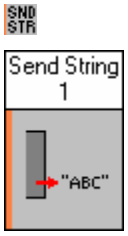
Short description

This tool sends a string to the specified I/O device.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool gets an ASCII string from the string buffer and sends it to the specified I/O device (see "APPENDIX A. ASCII table"). Depending on the type of the I/O device, the tool works in two modes – serial or TCP.

***ATTENTION**. Working with serial and TCP I/O devices requires special setup. Please read I/O device description in "*8.1. I/O devices*".*

.

The number of string bytes to send is determined by a terminating byte (0x0D or 0x00), found in the string buffer, or by the "Length" argument (what comes first). The receiver of the string could be a "Receive String" tool, working on another camera or a PC.

In serial mode the tool does not implement any handshake – it sends the string and a terminating byte equal to CR=0x0D. The send and receive tools must have synchronized transfer rates.

Results :

The tool has no results.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Start	Start index in string buffer (where the sent string is read from).
Length	Maximum length of the sent string when not terminated by 0 or CR.
Device	I/O device number in the range [0,255].

Results

The tool has no results.

Similar tools

Send result Sends values via I/O device

Usually combined with

“Receive String” tool, running on another camera or PC that communicates with your program.

Tips & Tricks

If it doesn’t seem to work in serial mode, please check other baud-rates (default 9600). You may use a terminal program for that purpose. The sent data will be readable on the terminal with the exception of terminating byte.

Additionally please make sure you did successfully reconfigure the serial interface as described in “*8.1. I/O devices*”.

Examples

Not indexed yet – sorry.

8.11. Receive Point-list

Group

I/O tools

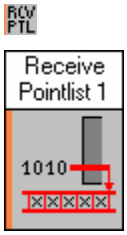
Short description

This tool receives a point-list from the specified I/O device.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool receives a point-list from the specified I/O device. Depending on the type of the I/O device, the tool works in two modes – serial or TCP.

***ATTENTION**. Working with serial and TCP I/O devices requires special setup. Please read I/O device description in "*8.1. I/O devices*".*

.

The sender of the point-list could be a "Send point-list" tool, working on another camera or a PC. In serial mode the data is received in packets, each with checksum. Damaged packets are retransmitted.

TCP mode:

Some tool arguments are ignored in TCP mode.

Results :

The tool returns the number of point-list items that were received.

WARNING: There is a general rule, which is observed in the generation of all VIMOS tool results. When a given tool returns an error result, a non-zero error code is associated with the result. Although the tool writes some new result value, other tools which have linked arguments to this error result, will use the last valid tool result, produced by the tool without error, and not the newly saved value.

All I/O "***receive***" tools return errors (timeout or others) when there is no incoming data from the remote I/O device. **Don't rely on** result values (for example: number of received bytes = 0) to check if the tool has received or not some data. **Always check tool errors first and then check data counts or other result values !**

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Transfer	Transfer mode: • ASCII – bytes are encoded as two hex symbols [0,1, ..9,A,B..F] • BINARY – bytes are transferred directly **Ignored in TCP mode.**
Wait	Wait for incoming point-list data for 350ms or don't wait
Start index	Start point-list index where to put the received data.
Device	I/O device number in the range [0,255].

Results

Result	Description
Section size	Point-list length (how many point-list entries were received)

Similar tools

Receive result	receives values via I/O device
Send result	send values via I/O device
Send image area	send a part of the image (Frame buffer) via I/O device
Send point-list	reverse operation – send part of the point-list via I/O device

Usually combined with

All tools that use point-list entries.

Tips & Tricks

If it doesn't seem to work in serial mode, please check other baud-rates (default 9600). Additionally please make sure you did successfully reconfigure the serial interface as described in "*8.1. I/O devices*".

Examples

Not indexed yet – sorry.

8.12. Send Point-list

Group

I/O tools

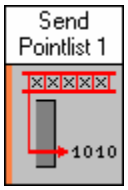
Short description

This tool sends a point-list to the specified I/O device.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool sends a point-list to the specified I/O device. Depending on the type of the I/O device, the tool works in two modes – serial or TCP.

***ATTENTION**. Working with serial and TCP I/O devices requires special setup. Please read I/O device description in "*8.1. I/O devices*".*

.

The receiver of the point-list could be a "Receive point-list" tool, working on another camera or a PC. In serial mode data is sent in packets, each with checksum. Damaged packets are retransmitted.

TCP mode:

Some tool arguments are ignored in TCP mode.

Results :

The tool returns a value that is zero for successful transfer or otherwise gives an error code.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Start index	Point-list start index (where to take the data from)
Section size	Point-list length (how many point-list items to transfer)
Transfer	Transfer mode: • ASCII – bytes are encoded as two hex symbols [0,1, ..9,A,B..F] • BINARY – bytes are transferred directly **Ignored in TCP mode.**
Device	I/O device number in the range [0,255].

Results

Result	Description
Success	Return result of sending – OK or Error

Similar tools

Send result	send values via I/O device
Send image area	send a part of the image (Frame buffer) via I/O device
Receive point-list	reverse operation – receive part of the point-list via I/O device
Receive result	receives values via I/O device

Usually combined with

All tools that produce point-list entries.

Tips & Tricks

If it doesn't seem to work in serial mode, please check other baud-rates (default 9600). You may use a terminal program for that purpose. If you use ASCII mode all the transferred data will be readable with the exception of the checksum. You could set the 'Start byte' to 0x0A (10) or 0x0D (13) to get a new line in the terminal for every new message. While readable the data will not be easily decodable without some thinking.

Additionally please make sure you did successfully reconfigure the serial interface as described in "*8.1. I/O devices*".

Examples

Not indexed yet – sorry.

8.13. Receive Image Area

Group

I/O tools

Short description

This tool receives an image area from the specified I/O device.

VIMOS kernel presentation

Editor icons

Description

General function:

The tool receives an image area from the specified I/O device. Depending on the type of the I/O device, the tool works in two modes – serial or TCP.

***ATTENTION**. Working with serial and TCP I/O devices requires special setup. Please read I/O device description in "*8.1. I/O devices*".*

In serial mode the tool works on PC only. In TCP mode the tool works both on PC and Ethernet camera VC20xxE.

The sender of the image is usually a "Send Image Area" tool, working on another camera or a PC. Other senders should transfer data in the format of the "Send Image Area" tool.

Serial mode:

In serial mode data is received in packets, each with checksum. Damaged packets are retransmitted. The send and receive tools must have synchronized transfer rates.

***ATTENTION**. The type of the transfer is specified by the "Transfer" argument of the "Send Image Area" tool - binary or ASCII. The ASCII mode is 2 times slower than binary mode, but is more reliable.*

Results :

The tool returns one float result - zero for successful transfer, otherwise an error code.

WARNING: There is a general rule, which is observed in the generation of all VIMOS tool results. When a given tool returns an error result, a non-zero error code is associated with the result. Although the tool writes some new result value, other tools which have linked arguments to this error result, will use the last valid tool result, produced by the tool without error, and not the newly saved value.

All I/O "***receive***" tools return errors (timeout or others) when there is no incoming data from the remote I/O device. **Don't rely on** result values (for example: number of received bytes = 0) to check if the tool has received or not some data. **Always check tool errors first and then check data counts or other result values !**

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Center	Center point of image rectangle where received image is stored.
Dest	Destination image buffer ('Frame buffer' or 'Freeze buffer')
Wait	Serial mode – wait mode for acknowledgement of each packet. TCP mode - wait mode for incoming image data. • **Don't wait** – don't wait • **Wait in mS** – wait "Wait in mS" milliseconds • **Wait forever** – wait forever
Wait in mS	Wait time in milliseconds used when the selected "Wait" option is "Wait in mS".
Device	I/O device number in the range [0,255].
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Success	Error code of transfer operation – OK or error.

Similar tools

Send result	send values via I/O device
Send point-list	send a part of the point-list via I/O device
Receive point-list	receive part of the point-list via I/O device
Receive result	receives values via I/O device
Send image area	send image area via I/O device

Usually combined with

In serial mode - “Send Image Area” tool, running on the camera that communicates with your program in simulator. In TCP mode - “Send Image Area” tool, running on arbitrary platform.

Tips & Tricks

Examples

Not indexed yet – sorry.

8.14. Send Image Area

Group

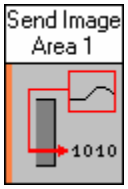
I/O tools

Short description

This tool sends an image area to the specified I/O device.

VIMOS kernel presentation

Editor icons



Description

General function:

This tool sends an image area to the specified I/O device. Depending on the type of the I/O device, the tool works in two modes – serial or TCP.

***ATTENTION**. Working with serial and TCP I/O devices requires special setup. Please read I/O device description in "*8.1. I/O devices*".*

In serial mode the tool works on camera only. In TCP mode the tool works both on PC and Ethernet camera VC20xxE.

The sender of the image is usually a "Send Image Area" tool, working on another camera or a PC. Other senders should transfer data in the format of this tool.

Serial mode:

In serial mode data is sent in packets, each with checksum. Damaged packets are retransmitted. Currently the image is sent in JPEG format only.

TCP mode:

Some tool arguments are ignored in TCP mode. Images are transferred in bitmap format as matrix of gray-level pixels.

Results:

The tool returns a value that is zero for successful transfer, otherwise returns an error code.

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Center	Center of image rectangle that has to be transferred.
Width	Width of image rectangle that has to be transferred.
Height	Height of image rectangle that has to be transferred.
Source	Source image buffer ('Frame buffer' or 'Freeze buffer').
Quality	Image quality for compression (higher quality means less compression and more data). **Ignored in TCP mode.**
Transfer	Transfer mode: • ASCII – bytes are encoded as two hex symbols [0,1, ..9,A,B..F] • BINARY – bytes are transferred directly **Ignored in TCP mode.**
Blocks	Size of transmission packets, 0 – all in one block. Non-zero blocks value enables wait and checksum options, otherwise they are ignored. **Ignored in TCP mode.**
Checksum	Enables/disables use of checksum for block transfer if wait is non-zero. **Ignored in TCP mode.**
Wait reply	Serial mode – wait time in ms for acknowledgement of each packet (0 = no acknowledgements). TCP mode – wait time in ms for reply from the "Receive image area" tool (0 = don't wait for reply).
Device	I/O device number in the range [0,255].
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Success	Return result of sending – OK or Error

Similar tools

Send result	send values via I/O device
Send point-list	send a part of the point-list via I/O device

Usually combined with

“Receive image area” tool working on another platform.

Tips & Tricks

If it doesn’t seem to work in serial mode, please check other baud rates (default 9600). Additionally please make sure you did successfully reconfigure the serial interface as described in “*8.1. I/O devices*”.

Examples

Not indexed yet – sorry.

8.15. Receive Binary

Group

I/O tools.

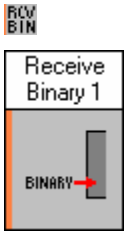
Short description

This tool receives a byte (if any) from the specified I/O device.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool receives a byte (if any) from the specified I/O device and saves the byte as tool result (-1: byte not available). The byte is received in binary format (raw data) without any handshake. The sender of the byte could be a "Send Binary" tool, working on another camera or a PC. Depending on the device configuration, the tool works in two modes – serial or TCP.

***ATTENTION**. Working with serial and TCP I/O devices requires special setup. Please read I/O device description in "*8.1. I/O devices*".*

Currently this tool is disabled on ADSP and sensor cameras.

In serial mode the sender and the receiver must have synchronized transfer rates.

Results :

The tool returns one float result, equal to the received byte [0,255] or –1 if byte is not available.

WARNING: There is a general rule, which is observed in the generation of all VIMOS tool results. When a given tool returns an error result, a non-zero error code is associated with the result. Although the tool writes some new result value, other tools which have linked arguments to this error result, will use the last valid tool result, produced by the tool without error, and not the newly saved value.

All I/O "***receive***" tools return errors (timeout or others) when there is no incoming data from the remote I/O device. **Don't rely on** result values (for example: number of received bytes = 0) to check if the tool has received or not some data. **Always check tool errors first and then check data counts or other result values !**

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Wait	Wait time in ms (0: no wait)
Device	I/O device number in the range [0,255].

Results

Result	Description
Received byte	Received byte [0,255] or –1 if byte is not available.

Similar tools

Receive string Receives string via the I/O device

Usually combined with

“Send Binary” tool, running on another camera or PC.

Tips & Tricks

You may receive sequence of bytes by executing “Receive binary” in a GOTO-LABEL loop.

If it doesn’t seem to work in serial mode, please check other baud-rates (default 9600). Additionally please make sure you did successfully reconfigure the serial interface as described in “*8.1. I/O devices*”.

Examples

Not indexed yet – sorry.

8.16. Send Binary

Group

I/O tools.

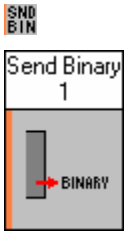
Short description

This tool sends binary data to specified I/O device.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool gets a sequence of bytes from the string buffer or a single byte from the "Byte" argument and sends the data in binary format (raw bytes) to the specified I/O device. The tool does not implement any handshake or transmission protocol. The receiver of the sent data could be a "Receive Binary" tool, working on another camera or a PC. Depending on the device configuration, the tool works in two modes – serial or TCP.

***ATTENTION**. Working with serial and TCP I/O devices requires special setup. Please read I/O device description in "*8.1. I/O devices*".*

Currently this tool is disabled on ADSP and sensor cameras.

Results :

The tool has one float result – transmission error code (0=OK).

Algorithm

Please just read the description chapter above.

Arguments

Argument	Description
Start	Start index in the string buffer where bytes are read from (Mode = Send string buffer").

Length	Number of string buffer bytes to send (Mode = “Send string buffer”).
Byte	Single byte to send (Mode = “Send byte”).
Mode	Send mode – “Send string buffer” or “Send byte”.
Device	I/O device number in the range [0,255].

Results

Result	Description
Error code	Transmission error code (0=OK).

Similar tools

Send string Sends string via the I/O device

Usually combined with

“Receive Binary” tool, running on another camera or PC.

Tips & Tricks

If it doesn’t seem to work in serial mode, please check other baud-rates (default 9600). Additionally please make sure you did successfully reconfigure the serial interface as described in “*8.1. I/O devices*”.

Examples

Not indexed yet – sorry.

8.17. Create Server Device

Group

I/O tools.

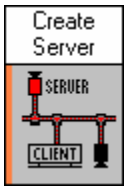
Short description

This tool creates (configures) server I/O device.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool configures the selected device as a TCP server. The tool does not fulfill any connections, it just setups the I/O device. The actual connection is done automatically in run mode. VIMOS checks for connection requests from remote clients and accepts a request if the remote peer has IP address, equal to the IP address specified by the tool.

If the selected device has been already configured as a TCP server or client, the tool ignores the previous settings, disconnects the device and configures it as a TCP server device. The tool does not disconnect a connected device when executed with the same arguments.

The tool can be executed regularly in the main system loop to check the connection state of the device. The tool returns error –2107 if the device is not connected and 0 if connected.

***ATTENTION**. Stopping of run mode and entering into edit mode disconnects all TCP devices and closes the created sockets. Disconnected TCP devices become serial.*

***ATTENTION** Several connection requests from one client (i.e. requests with identical IP addresses) could be accepted on camera not in the order the client issues them. To accept requests in the same order, ensure a wait time above 300-400 ms between consecutive connect requests from the client.*

Results :

The tool returns one float result – the connection state of the device: 0 = connected, non-zero = not connected or I/O error.

Algorithm

Please just read the description above.

Arguments

Argument	Description
Device	I/O device number in the range [0,255].
IP address (byte 0)	IP address of remote client – byte 0 (first byte).
IP address (byte 1)	IP address of remote client – byte 1.
IP address (byte 2)	IP address of remote client – byte 2.
IP address (byte 3)	IP address of remote client – byte 3 (last byte).
Port	TCP port used for listening (default = 2000, currently can't be modified).

Results

Result	Description
Result	Connection state (0=connected).

Similar tools

Client Device Connect Connects device as TCP client.

Usually combined with

I/O tools working in TCP mode with server device configured by this tool.

Tips & Tricks

To disconnect a connected server or client device, reconfigure it as server or client and set a non-existing IP address.

Examples

Not indexed yet – sorry.

8.18. Client Device Connect

Group

I/O tools.

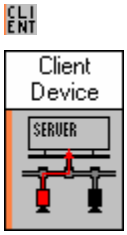
Short description

This tool connects a client I/O device.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool connects the selected device as TCP client to remote server, specified by IP address and port. The tool waits maximum "**Timeout**" ms to make connection before exit (timeout=0: no wait).

If the selected device has been already configured as TCP server or client, the tool ignores the previous settings, disconnects the device and re-connects it as TCP client to a server with specified IP address and port. The tool does not disconnect a connected device on next execution with the same arguments.

The tool can be executed regularly in the main system loop to check the connection state of the device. The tool returns error –2107 if the device is not connected and 0 if connected.

***ATTENTION**. Stopping of run mode and entering into edit mode disconnects all TCP devices and closes the created sockets. Disconnected TCP devices become serial.*

Results :

The tool returns one float result – the connection state of the device: 0 = connected, non-zero = not connected or I/O error.

Algorithm

Please just read the description above.

Arguments

Argument	Description
Device	I/O device number in the range [0,255]
IP address (byte 0)	IP address of remote server – byte 0 (first byte).
IP address (byte 1)	IP address of remote server – byte 1.
IP address (byte 2)	IP address of remote server – byte 2.
IP address (byte 3)	IP address of remote server – byte 3 (last byte).
Port	TCP port of remote server (default = 2000, currently can't be modified).
Timeout	Connection wait time in milliseconds (0=no wait).

Results

Result	Description
Result	Connection state (0=connected).

Similar tools

Create Server Device Creates TCP server device.

Usually combined with

I/O tools working in TCP mode with client device connected by this tool.

Tips & Tricks

To disconnect a connected server or client device reconfigure it as server or client and set non-existing IP address.

Examples

Not indexed yet – sorry.

8.19. Open Serial Device

Group

I/O tools.

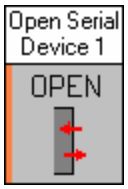
Short description

This tool opens a serial device for I/O transfer via the RS232/RS422 port of an Ethernet camera like VC40xxE and FA45.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function :

The tool opens a serial device for I/O transfer via an RS232/RS422 port of a camera, which has both Ethernet and serial port (VC40xxE, FA45). On all other camera models and on PC the tool is NOP (no operation). The tool sets a new baud rate of the serial port from the **Baud_rate** argument (if not already set). The default baud rate is taken from the properties of the VIMOS program ("I/O baud rate"). Working with RS422 devices does not require "Other I/O device" to be selected. The **Serial_port** argument specifies a serial port number from 0 to 7 (currently ignored). Serial port number 8 is used to free the current device and restore it its default state – the ***default*** serial device.

ATTENTION. *Remember that this serial device is different from the* ***default*** *serial device, which is Telnet TCP port 23 (the port which is used to communicate with the shell of the Ethernet cameras). An I/O device always works as default serial device unless it is initialized by this tool or by a TCP server/client tool.*

Results :

The tool returns one float result, equal to the error code of the operation (0=OK).

Algorithm

Please just read the description above.

Arguments

Argument	Description

Device	I/O device number in the range [0,255]
Serial_port	Serial port number in the range [0,7], 8=restore to default serial device. **Currently ignored.**
Baud_rate	Baud rate of the serial port:: • default (use baud rate from VIMOS configuration) • 1200 – 115200 (set this baud rate if not already set)
Three_state	Three-state operation. **Currently ignored.**

Results

Result	Description
Result	Tool operation error (0=OK)

Similar tools

Create Server Device	Creates TCP server device.
Client Device Connect	Connects a TCP client device.

Usually combined with

I/O tools working with the opened serial device.

Tips & Tricks

Examples

Not indexed yet – sorry.

9. Program flow

Normally the VIMOS program contains a sequence of tools and these tools are executed one after another from the beginning of the program to its end. In some cases it may be necessary to change execution sequence depending on the situation. As an example you may want to display an error message if a certain result is not within the green.

At present there are three main ways to control the program flow:

- The first way is to use ‘IF- constructs’.
- The second way is to use ‘GOTO-LABEL combinations’. Usually to ‘GOTO’ itself is used within an ‘IF-construct’.
- The third way is to use subroutines. Currently creation of user-program subroutines is possible only by the editor.

I’m using the term ‘IF-constructs’ here because by using the IF operators, you are able to build hierarchical structures of these operators that can become complex. Within the Editor there is only one IF-ELSE operator, which is able to accept complex structures. Implementing such an operator is not possible in the camera so within the ‘VIMOS-kernel’ there is a group of such operators that allow in their combination to create complex structures too. This difference between Editor and Kernel (Camera, Simulator) makes explanations a bit tricky.

Most of the explanations below refer to ‘IF-constructs’ in VIMOS kernel. Please read the editor manual for more details about branches in editor programs.

The combination of ‘GOTO’ and ‘LABEL’ is much simpler to handle and a real programming classic ☺.

Editor toolbar of this group of tools:

9.1. Get Cycles

Group

Program flow.

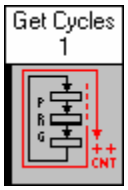
Short description

The tool returns the number of executed user-program cycles.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool returns the number of user-program cycles completed since the start of the run-mode. The system resets this number to zero each time it enters run-mode. In edit-mode this tool always returns zero. The tool has no arguments and configuration dialog in VIMOS kernel.

Results:

The tool returns one float result - current program cycle.

Algorithm

Please read the description chapter above.

Arguments

The tool has no arguments.

Results

Result	Description
Program cycle	Number of current user-program execution cycle.

Similar tools

Usually combined with

Tips & Tricks

Use the tool to perform some initializations in cycle 0 of the user-program. Example:

```
Get cycles [R1]          /* Get program cycle in R1       */
If [<,R1,1]              /* If R1 < 1 (cycle 0) then:     */
. . .                    /* Program initialization        */
Endif
. . .                    /* Tools, executed in all cycles */
```

Such initialization is especially important when you need to execute some tools only once, for example tools which read or write data from/to flash files.

Examples

Not indexed yet – sorry.

9.2. IF command

Group

Program flow.

Short description

Introduces the first condition of a compound conditional statement.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons

Description

General function:

Introduces the first condition of a compound conditional statement. Each elementary condition is a comparison of the argument **Result** with the immediate value of the argument **Value**. The source of the **Result** operand is a tool result the argument is linked to. The conditional statement contains compound conditions, each one of them being a sequence of elementary conditions.

***ATTENTION**. Remember that when a tool fails, it sets result's error code only but keeps result value unchanged. As a general rule, if you want to check result error code, do it before checking the result value.*

In general, a conditional sequence looks like this (the ELSEIF-parts and the ELSE-part are optional):

```
IF-part
ELSEIF-part1
ELSEIF-part2
…
ELSE-part
ENDIF
```

The **IF**-part begins with an IF command and is followed by an optional sequence of ANDIF and ORIF commands. All these IF commands produce the compound condition. It is followed by any sequence of arbitrary elements.

The **ELSEIF**-part begins with an ELSEIF command. Just like the IF-part, it could be followed by some combination of ANDIF and ORIF commands. Again the compound condition is followed by any kind of elements.

The **ELSE**-part starts with an ELSE command. It is followed by any sequence of arbitrary elements. The **ENDIF** command ends the conditional sequence.

Note: The ANDIF command has higher precedence than ORIF. For example, the compound condition:

```
IF cond1
ORIF cond2
ANDIF cond3
```

is equivalent to:

```
cond1 OR (cond2 AND cond3).
```

During execution, the compound conditions are evaluated sequentially, starting form the first one (IF-part). If the condition is satisfied, the content of the respective part is processed and all the rest parts down to ENDIF are skipped. If the condition is not satisfied, the respective part is skipped and the next one is inspected. If none of the conditions are satisfied, only the ELSE-part is processed. In run-mode skipped elements are neither executed nor drawn. Conditional sequences could be nested to any level. The tool has no results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Compare	Specifies what to compare - a result value or an error code associated with the result. If a tool encounters an error during its execution, it stores respective error code into its result and leaves the result value unchanged.
Point axis	Selector for X-axis or Y-axis when the compared result is of type POINT. Ignored for non-point results.
Result	An argument, which is compared with the immediate value. The argument must be linked to a tool result.
Operator	Comparison operator: <, <=, ==, !=, >=, >
Value	Immediate floating-point value, compared with result

Results

This tool has no results.

Similar tools

All conditional tools from this group.

Usually combined with

ANDIF, ORIF, ELSE, ELSIF, ENDIF, GOTO and LABEL tools.

Tips & Tricks

Examples

Not indexed yet – sorry.

9.3. ANDIF command

Group

Program flow.

Short description

The tool adds AND condition to compound conditional statement, started previously by IF or ELSEIF tools.

VIMOS kernel presentation

The tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons

Description

General function:

The tool adds next condition to a compound conditional statement. A Boolean AND operator combines this condition with previous conditions. The compound conditional statement must start with an IF or ELSEIF command. The ANDIF commands have higher precedence than ORIF commands. The tool has no results. Refer to description of IF tool for more details.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Compare	Specifies what to compare - a result value or an error code associated with the result. If a tool encounters an error during its execution, it stores respective error code into its result and leaves the result value unchanged.
Point axis	Selector for X-axis or Y-axis when the compared result is of type POINT. Ignored for non-point results.
Result	An argument, which is compared with the immediate value. The argument must be linked to a tool result.
Operator	Comparison operator: <, <=, ==, !=, >=, >
Value	Immediate floating-point value, compared with result

Results

This tool has no results.

Similar tools

All conditional tools from this group.

Usually combined with

IF, ORIF, ELSE, ELSIF, ENDIF, GOTO and LABEL tools.

Tips & Tricks

Examples

Not indexed yet – sorry.

9.4. ORIF command

Group

Program flow.

Short description

The tool adds OR condition to compound conditional statement, started previously by IF or ELSEIF tools.

VIMOS kernel presentation

The tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons

Description

General function:

The tool adds next condition to a compound conditional statement. A Boolean OR operator combines this condition with previous conditions. The compound conditional statement must start with an IF or ELSEIF command. The ORIF commands have lower precedence than ANDIF commands. The tool has no results. Refer to description of IF tool for more details.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Compare	Specifies what to compare - a result value or an error code associated with the result. If a tool encounters an error during its execution, it stores respective error code into its result and leaves the result value unchanged.
Point axis	Selector for X-axis or Y-axis when the compared result is of type POINT. Ignored for non-point results.
Result	An argument, which is compared with the immediate value. The argument must be linked to a tool result.
Operator	Comparison operator: <, <=, ==, !=, >=, >
Value	Immediate floating-point value, compared with result

Results

This tool has no results.

Similar tools

All conditional tools from this group.

Usually combined with

IF, ANDIF, ELSE, ELSIF, ENDIF, GOTO and LABEL tools.

Tips & Tricks

Examples

Not indexed yet – sorry.

9.5. ELSEIF command

Group

Program flow.

Short description

The tool begins ELSEIF part of a compound conditional statement, started previously by the IF tool.

VIMOS kernel presentation

The tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons

Description

General function:

The tool begins ELSEIF-part of compound a conditional statement. This part is executed when the previous IF or ELSEIF conditions are false. Several ELSEIF parts may follow initial IF part. The tool has no results. Refer to description of IF tool for more details.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Compare	Specifies what to compare - a result value or an error code associated with the result. If a tool encounters an error during its execution, it stores respective error code into its result and leaves the result value unchanged.
Point axis	Selector for X-axis or Y-axis when the compared result is of type POINT. Ignored for non-point results.
Result	An argument, which is compared with the immediate value. The argument must be linked to a tool result.
Operator	Comparison operator: <, <=, ==, !=, >=, >
Value	Immediate floating-point value, compared with result

Results

This tool has no results.

Similar tools

All conditional tools from this group.

Usually combined with

IF, ANDIF, ORIF, ELSE, ENDIF, GOTO and LABEL tools.

Tips & Tricks

Examples

Not indexed yet – sorry.

9.6. ELSE command

Group

Program flow.

Short description

The tool adds ELSE part of a compound conditional statement, started previously by the IF tool.

VIMOS kernel presentation

The tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons

Description

General function:

The tool adds the ELSE-part of a compound conditional statement. This part is executed if all previous conditions evaluate to FALSE. This tool has no arguments - therefore it has no configuration dialog box in VIMOS kernel. The tool has no results. Refer to description of IF tool for more details.

Algorithm

Please read the description chapter above.

Arguments

The tool has no arguments.

Results

This tool has no results.

Similar tools

All conditional tools from this group.

Usually combined with

IF, ANDIF, ORIF, ELSEIF, ENDIF, GOTO and LABEL tools.

Tips & Tricks

Examples

Not indexed yet – sorry.

9.7. ENDIF command

Group

Program flow.

Short description

The tool terminates a compound conditional statement, started previously by the IF tool.

VIMOS kernel presentation

The tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons

Description

General function:

The tool ends a compound conditional statement. The tool has no arguments and results.

Algorithm

Please read the description chapter above.

Arguments

The tool has no arguments.

Results

This tool has no results.

Similar tools

All conditional tools from this group.

Usually combined with

IF, ANDIF, ORIF, ELSE, ELSEIF, GOTO and LABEL tools.

Tips & Tricks

Examples

Not indexed yet – sorry.

9.8. GOTO command

Group

Program flow.

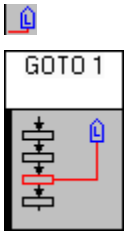
Short description

The tool jumps to a user-program location specified by a LABEL tool.

VIMOS kernel presentation

The tool has no graphical representation.

Editor icons



Description

General function:

The GOTO and LABEL tools perform unconditional jumps inside the user program. The GOTO tool jumps to a user-program location, specified by a LABEL tool. The label identifier (number) is specified by the “Goto” argument. If a corresponding LABEL command is not found in the user program, the GOTO command is equivalent to NOP. In case of several identical LABEL commands, the control is passed to the first one (the LABEL command with lowest program index).

If LABEL is behind GOTO (i.e. with greater user-program index), the GOTO-LABEL sequence is equivalent to IF-ENDIF with false condition:

```
. . .
Goto [0]
. . .
Label [0]
. . .
```

If LABEL is before GOTO, a loop inside the user-program occurs. To avoid infinite program loop, include GOTO in a conditional statement IF-ENDIF, which counts the number of loop cycles.

Example:

```
. . .
Set Cnt [1,10]          /* Set counter C1 = 10                    */
Label [0]               /* LABEL0                                 */
. . .                   /* Tool executed in loop                  */
Dec Cnt [1]             /* Decrement counter C1                   */
If [>,C1,0]             /* If C1 > 0, then execute next GOTO      */
Goto [0]                /* GOTO LABEL0                            */
ENDIF
. . .                   /* Next tool after loop                   */
```

The example above loops 10 times the tools, closed between LABEL and GOTO. Note that this loop construction performs minimum 1 loop cycle.

***ATTENTION**. Be careful not to place a LABEL command inside IF-ELSE-ENDIF sequence. This may cause unpredictable behavior of the user program.*

Don't put GUI tools in such kind of loop as shown by the example. Correct processing of mouse and touch-screen commands is provided only by complete cycles of the whole user-program.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Goto	Number of LABEL tool, which specifies the target jump location. Next executed tool is the tool which follows the LABEL tool (the LABEL tool actually does nothing).

Results

This tool has no results.

Similar tools

LABEL tool, conditional commands.

Usually combined with

LABEL tool.

Tips & Tricks

Examples

Not indexed yet – sorry.

9.9. LABEL command

Group

Program flow.

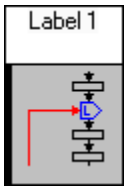
Short description

The tool specifies target user-program location for jump, done by the GOTO tool.

VIMOS kernel presentation

The tool has no graphical representation.

Editor icons



Description

General function:

The LABEL tool specifies a target user-program location, which is executed after a GOTO jump. The “Label” argument of the tool should specify label number (identifier), which is equal to the label of the respective GOTO tool. The tool has no results. See description of GOTO tool for more details.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Label	Number (identifier) of a label, which specifies target jump location. Next executed tool is the tool which follows the LABEL tool (the LABEL tool performs actually NOP operation).

Results

This tool has no results.

Similar tools

GOTO tool, conditional commands.

Usually combined with

GOTO tool.

Tips & Tricks

Examples

Not indexed yet – sorry.

9.10. Load User Program

Group

Program flow.

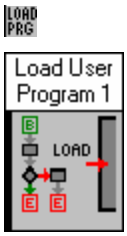
Short description

The tool loads and starts execution a new user-program file. There is no automatic return to the calling user-program.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool replaces current user-program (UP) by another one. The new UP is loaded from a file, specified by the "File name" argument and overwrites current UP. Side effects of this operation are:

- All tools in current UP are lost. Remember to save current UP before starting it if you have created the program in VIMOS kernel.
- Float, point and angle results of the current UP are lost. Counters (C) are preserved.
- The new UP is executed, starting from the beginning.

***ATTENTION**. This tool does not make a subroutine call – the new UP simply replaces current UP. Use subroutines created by Editor if you want to return to the calling UP. Calling subroutines does load new UP file - subroutines are contained in current UP file.*

No parameters can be passed to the called programs except counters. As there is no function-style return to old UP no return information is kept. You may load the old UP, but it will start execution from the beginning. This tool works in **run-mode** only.

If the tool fails, VIMOS kernel aborts execution with error message 10003 (the UP file does not exist). The tool has no results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
File name	Name of user-program file to load and execute.

Results

This tool has no results.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

9.11. Start Exec

Group

Program flow.

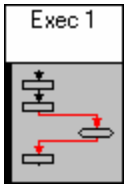
Short description

The tool executes an external user-coded module.

VIMOS kernel presentation

This tool has not a fixed graphical representation.

Editor icons



Description

General function:

The tool starts external execution module with predefined name (exec1 to exec9), coded by the user. It gives the possibility to execute an external plug-in tool by the VIMOS kernel. The tool passes 10 input arguments to the module (Arg.1 to Arg.10) and receives 20 results of different types. Each input argument can be linked to a tool result of arbitrary type **R**, **P**, **A** or **C**. Instructions and template code for creation of external plug-in modules can be delivered upon request (the argument and result passing rules are described in the header file **exe_pack.h**).

The external modules can draw in the overlay picture. Note: Overlay drawing currently is supported by the PC simulator only (external modules are created in the form of DLL libraries).

Results:

The tool returns 20 results: 5 float results **R,** 5 point results **P**, 5 angle results **A** and 5 counters **C**. The tool error codes have values **`5100 + RC`**, where **`RC`** is a non-zero error code, returned by the external module.

***ATTENTION**. Counters are special types of results. When a "Start Exec" tool is added to the user program, next 5 free counters, not reserved for other tool results, are assigned as results of the tool. If some counter should be preserved after the execution of the external module, link the counter to some input argument. When writing module code, store the input counter argument as module result (type **C)**.*

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Arg.1	First module argument – can be linked to results **R**, **P**, **A** and **C**.
Arg.2	Second module argument – can be linked to results **R**, **P**, **A** and **C**.
. . . . .	. . . . .
Arg.10	Last module argument - can be linked to results **R**, **P**, **A** and **C**.
File name	Name of user exec module: exec1 to exec9.

Results

Result	Description
Float results	5 float-results **R**.
Point results	5 point-results **P**.
Angle results	5 angle-results **A**.
Counter results	5 counter-results **C**.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

9.12. Exit User Program

Group

Program flow.

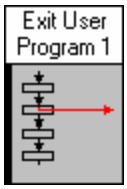
Short description

This tool stops user-program execution and enters into edit mode of VIMOS kernel or stops kernel execution (exits to operating system on camera or exits from kernel simulation in PC).

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

This tool stops user-program execution. The input argument "Exit to" selects the type of exit - exit to edit mode of VIMOS kernel or exit to VCRT (the operating system of the camera). In PC simulator the VCRT option exits from kernel simulation, but does not exit to PC operating system. The tool has no results.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Exit to:	Type of exit: • **Edit Mode** – stop user-program execution and enter edit mode of VIMOS kernel • **VCRT** – stop execution of VIMOS kernel and exit to camera operating system (exit kernel simulation in PC simulator).

Results

This tool has no results.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

10. Object analysis tools

This group of tools detects objects and finds object features by working with BLOBs. The term **BLOB** is abbreviation for "**B**inary **L**arge **OB**ject". The basic idea of the BLOB processing is to divide the pixels in the input image in two groups - foreground pixels (usually marked by 1 or -1) and background pixels (usually marked by 0). Sets of connected foreground pixels, which are surrounded by background pixels, represent the objects. Such objects have features like area (number of pixels), mass center, orientation, shape and so on. Further in this chapter BLOBs are also called ***objects***.

The tools in this group should be executed in two stages:

Stage 1: Initial object detection and generation of *object data base* (ODB) buffer.

This stage is realized by the "**Get Objects**" tool. This tool works on a region of interest (ROI) area, defined by a non-rotated rectangle in the input image. The tool creates an ODB buffer with object description data and calculates some basic object features like area and mass center. You can work with multiple ODB buffers, created from different images. ODB buffers are identified by indexes in the range [0,1023].

Stage 2: Calculation and extraction of advanced object features.

The next tools use ODB as input/output buffer. They calculate object features, store feature values in ODB and extract feature values from ODB to different VIMOS resources.

- The "**Get Object Features**" tool calculates selected features of one object and returns the feature values as tool results.
- The "**Get Object Features 2**" tool calculates requested features and saves feature values in the VIMOS point-list buffer. This tool can process all objects in ODB or a subset of objects, selected by a list with object indexes.

Once calculated for a given object, the features are remembered in ODB. Next request for the same features do not recalculate features – they just read values from ODB. Remember that in unconnected mode you can get basic features only (see the "Get Objects" tool). Currently the two tools do not support calculation of advanced features for unconnected objects.

***ATTENTION**. The tools from this group work on PC and TI camera. The tools "Get Object Features" and "Get Object Features 2" return error BLOB_R_INV_FEATURE (9111) on attempt to calculate advanced features in unconnected mode.*

Here is a brief summary of the calculated object features:

- Position and dimensions of the non-rotated rectangle, which encloses the object.
- Object area (number of object pixels).
- Object mass center.
- Average object brightness = SUM(gray-level object pixels) / area.
- Length of object's external contour.
- Number of object holes.
- Minimum and maximum distance from the mass center to a contour point.
- Major (greater) radius and minor (smaller) radius of equivalent ellipse. The center of the equivalent ellipse is the mass center.
- Rotation angle of equivalent ellipse.
- Area of object's convex hull (number of pixels).
- Greater extent (width) and smaller extent (height) of minimum area enclosing rectangle – a rotated rectangle which encloses the object.
- Rotation angle of minimum area enclosing rectangle
- Shape features – compactness, circularity, anisometry, bulkiness and convexity.

Editor toolbar of this group of tools:


Object ...
PAR

10.1. Get Objects

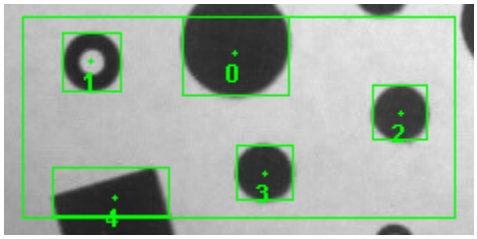
Group

Object analysis tools.

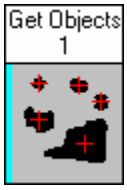
Short description

The tool performs binarization of the input image with two thresholds, finds objects, performs object labeling and creates object data base (ODB) buffer. The tool calculates some basic features like area and mass center.

VIMOS kernel presentation



Editor icons



Description

General function:

The tool works on input non-rotated rectangle area (ROI), defined by the arguments **Center**, **Width** and **Height**. It binarizes the input gray-level image with lower and upper binarization thresholds **Lo_thresh** and **Up_thresh**:

Lo_thresh <= pixels <= **Up_thresh**	Object area
pixels < **Lo_thresh** or pixels > **Up_thresh**	Background area

The tool detects BLOB objects, performs object labeling (segmentation) and creates a result "object-data base" (ODB) buffer with index **ODB_id**. If an ODB buffer with this index exists, then the tool discards (frees) the buffer and allocates a new ODB buffer.

The tool performs object filtering – objects, which do not satisfy certain conditions, are not stored into ODB. The arguments **Min_area** and **Max_area** exclude objects with area outside the range [**Min_area**, **Max_area**]:

area < **Min_area** or area > **Max_area**	Exclude objects with such area
Min_area, **Max_area** = [0,0]	Ignore area object filtering

The argument **Filt_mask** excludes objects with given position relative to the image boundaries.

The tool can work in destructive mode (see the argument **Destructive**) by setting background and foreground (object) image pixels to colors, specified by **Bgnd_clr** and **Obj_clr**. We recommend working in destructive mode while adjusting the binarization thresholds for the processed images, because you can see how the tool separates objects from background – the default colors are white objects on black background.

In ***unconnected*** mode the tool treats all foreground image pixels as one object. The created ODB buffer contains one object with index 0. The basic features described below are valid in unconnected mode. The advanced features, calculated by the tools from this group are not available. The easiest way to get unconnected features is to use the "**Get Object Features**" tool with disabled calculation flags - see the description of the tool for more details.

The tool allocates an ODB buffer with size **ODB_size** bytes. The size of the ODB buffer depends on the type and the size of the input image. A zero value of **ODB_size** instructs the tool to allocate ODB with approximate size, based on an estimation of the input image. This automatic size calculation works in most cases, but it is possible to receive ODB overflow errors 9102, 9104, 9106 and 9110 for some special input images with large number of objects and/or too long contours. In this case you should use a non-zero **ODB_size** value, which is greater than the automatic size. The tool results **ODB_size** and **Tot_ODB_size** show the actual ODB size and the total allocated ODB memory in bytes.

One approximate formula for the manual calculation of the ODB size is:
ODB size = 20000 + (expected number of objects) * 300
Thus a size of 300000 bytes for an image with about 1000 objects looks a reasonable value.

Note: The approach with approximate ODB sizes has been accepted because the accurate size calculation adds overhead execution time and requires calculation of features, which may be redundant for your application.

***ATTENTION**. Remember that the feature calculation tools "Get Object Features" and "Get Object Features 2" need additional free space in ODB for contour and other data.*

The total amount of allocated ODB memory is limited by the size of the available free heap DRAM of the PC and the TI camera.

The tool calculates and stores in ODB the following basic object features, which can be extracted by the "Get Object Features" and "Get Object Features 2" tools:

Mass center	Center of mass
Rect center	Center of non-rotated enclosing rectangle
Contour point	Beginning contour point (a point from the outer object's contour)
Width	Width of object's non-rotated enclosing rectangle
Height	Height of object's non-rotated enclosing rectangle
X_min	X-coordinate of top/left corner of non-rotated enclosing rectangle
Y_min	Y-coordinate of top/left corner of non-rotated enclosing rectangle
Area	Object area in pixels
Avg_brightness	Average object brightness = sum(gray-level object pixels) / area

The tool can draw some features – mass centers, object numbers and non-rotated enclosing rectangles. The same features can be drawn by the feature calculation and extraction tools "Get Object Features" and "Get Object Features 2". There is no sense to enable drawing of these features in several tools - choose one only.

Results:

The tool returns 4 float results – ODB identifier, number of objects in ODB, actual size of ODB data and total allocated ODB memory in bytes.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Center	Center rectangle point
Width	Rectangle width
Height	Rectangle height
Lo_thresh	Lower binarization threshold [0,255]
Up_thresh	Upper binarization threshold [0,255] (>= **Lo_thresh**, set to 255 if < **Lo_thresh**)
Min_area	Min object area for object filtering
Max_area	Max object area for object filtering: **Min_area**, **Max_area** = [0,0] : ignore area filtering **Min_area** > **Max_area** : ignore **Max_area**
Filt_mask	Filter mask which excludes objects with given position relative to the image boundaries: • Filt_mask = 0 : include all object types • bit 0 : exclude left boundary objects (add 1) • bit 1 : exclude right boundary objects (add 2) • bit 2 : exclude top boundary objects (add 4) • bit 3 : exclude bottom boundary objects (add 8) • bit 4 : exclude internal (non-boundary) objects (add 16)
Destructive	Enable destructive mode - set background and object pixels in the image to **Bgnd_clr** and **Obj_clr**.
Connected	Specifies object connection mode : • 0 : unconnected mode – all foreground pixels are treated as one possibly unconnected object • 1 : search 8-connected objects (default) • 2 : search 4-connected objects
Bgnd_clr	Background color stored into image in destructive mode.
Obj_clr	Object color stored into image in destructive mode - default colors: **Bgnd_clr** = 0, **Obj_clr** = 255.
Draw_clr	Tool drawing color on success
Err_clr	Tool drawing color on error
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only
Disp_mode	Feature display mode: • bit 0 : draw object numbers • bit 1 : draw mass centers • bit 2 : draw non-rotated enclosing rectangles
ODB_size	Size of allocated ODB buffer in bytes (0=auto)

ODB_id	ODB identifier [0,1023]

Results

Result	Description
ODB_id	ODB identifier , equal to the argument ODB_id
ODB_cnt	Number of objects in ODB
ODB_size	Actual ODB size in bytes (excluding the free space, needed for next calculations - contour data,...)
ODB_tot_size	Total allocated ODB memory in bytes

Similar tools

Find BLOBs.

Usually combined with

The “Get Object Features” and “Get Object Features 2” tools, which calculate and extract object features.

Tips & Tricks

Examples

Not indexed yet – sorry.

10.2. Get Object Features

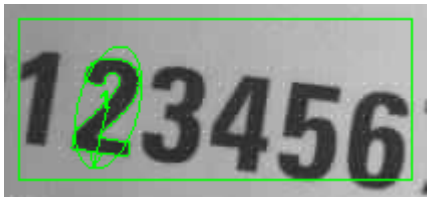
Group

Object analysis tools.

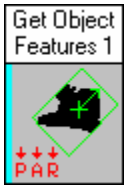
Short description

The tool calculates selected features for one object and returns all feature values as tool results.

VIMOS kernel presentation



Editor icons



Description

General function:

The tool uses as input/output buffer an "object data base" (ODB) buffer with index **ODB_id**, created by the "**Get Objects**" tool. The tool calculates the features of one single object, specified by the **Obj_id** argument. The tool returns all ***float***, ***point*** and ***angle*** object features as tool results. If you need object contour points, then you should use the "**Get Object Features 2**" tool. In case of invalid or missing ODB buffer the tool returns error BLOB_R_INVALID_ODB (9118). In case of invalid object index the tool returns error BLOB_R_INV_OBJ_IDX (9108).

The tool does not calculate automatically all object features. The arguments **Calc_contour**, **Calc_ellipse** and **Calc_rect** specify which group of features must be calculated. If a feature is not calculated, then the tool returns a respective result with error BLOB_R_FEATURE_NOCALC (9113). If a requested group needs another group, the second group is also calculated:

- **Basic group of features**. These features are already calculated by "Get Objects" and you don't need to enable any calculation flags. The tool returns the basic features in the following results: **Mass_center, Rect_center, Contour_point, X_cent_r, Y_cent_r, Width, Height, X_min, Y_min, Area, X_cent, Y_cent** and **Avg_brightness**.
- **Calc_contour.** Calculates contour features and sets the results **Cont_len, Hole_cnt, Min_dist, Max_dist, Compactness** and **Circularity**. Does not force calculation of other groups.

- **Calc_ellipse.** Calculates equivalent ellipse features and sets the results **Major_rad, Minor_rad, Ellip_angle, Anisometry** and **Bulkiness**. Does not force calculation of other groups. If the contours are calculated, then the ellipse angle is in the range [0,2PI], else in [0,PI].
- **Calc_rect.** Calculates min area rectangle features and sets the results **Area_convex, Rect_dx, Rect_dy, Rect_angle** and **Convexity**. Forces calculations of contours.

Once calculated, the features of a given object remain stored in ODB. Next calls of the tool do not recalculate features – the tool just reads values from ODB. If a given group of features for this object has been already calculated, you will receive valid results even if you have not enabled the calculation of that group.

***ATTENTION**. Please read the description of the "Get Object Features 2" tool to learn more about features, feature groups and forced calculations.*

.

If the input ODB buffer has been created by "Get Objects" in unconnected mode, then you can get the basic features of one unconnected object with **index 0** in the results described above.

Tool drawing

The tool draws features with adequate graphical representations, which are selected by the **Disp_mode** argument. Uncalculated features are not drawn. The tool draws features of the selected object only.

Results:

The tool returns 28 results:

- 3 point results **(P)** – mass center, center of the non-rotated enclosing rectangle and beginning contour point
- 2 angle results **(A)** –ellipse angle and min area rectangle angle
- 23 float results **(R)** – feature values

Note that the point features "**Mass_center**" and "**Rect_center**" are available in two forms in different results – as points and as separate X/Y coordinates.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
ODB_id	ODB identifier [0,1023]
Obj_id	Object index, used to select 1 ODB object
Cont_limit	Contour limit used in the calculation of the object contour features. Specifies max number of detected contour points in one contour before the tool aborts execution with error 9119. **0 = don't check**
Calc_contour	Enables calculation of contour features.
Calc_ellipse	Enables calculation of ellipse features.
Calc_rect	Enables calculation of min area rectangle features.

Draw_clr	Tool drawing color on success
Err_clr	Tool drawing color on error
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only
Disp_mode	Feature display mode. Add the number 1,2,4,… to enable the display of the respective feature: • bit 0 : draw object number (add 1) • bit 1 : draw cross marker of mass center (add 2) • bit 2 : draw non-rotated enclosing rectangle (add 4) • bit 3 : draw object contour (add 8) • bit 4 : draw equivalent ellipse (add 16) • bit 5 : draw min area rotated enclosing rectangle (add 32)

Results

Result	Description
Mass_center	**(P)** Center of mass
Rect_center	**(P)** Center of non-rotated enclosing rectangle
Contour_point	**(P)** Beginning contour point from outer object's contour
X_cent_r	**(R)** X-coordinate of center of non-rotated enclosing rectangle
Y_cent_r	**(R)** Y-coordinate of center of non-rotated enclosing rectangle
Width	**(R)** Width of object's non-rotated enclosing rectangle
Height	**(R)** Height of object's non-rotated enclosing rectangle
X_min	**(R)** X-coordinate of top/left corner of non-rotated rectangle
Y_min	**(R)** Y-coordinate of top/left corner of non-rotated rectangle
Area	**(R)** Object area in pixels
X_cent	**(R)** X-coordinate of object's mass center
Y_cent	**(R)** Y-coordinate of object's mass center
Avg_brightness	**(R)** Average object brightness = sum(gray-level object pixels) / area
Cont_len	**(R)** Length of outer object's contour
Hole_cnt	**(R)** Number of object holes
Min_dist	**(R)** Min distance from the mass center to a contour point from the outer object contour
Max_dist	**(R)** Max distance from the mass center to a contour point from the outer object contour
Compactness	**(R)** Object compactness
Circularity	**(R)** Object circularity
Major_rad	**(R)** Greater radius of equivalent ellipse
Minor_rad	**(R)** Smaller radius of equivalent ellipse
Ellip_angle	**(A)** Rotation angle of equivalent ellipse [0,2PI]
Anisometry	**(R)** Object anisometry (elongatedness)

Bulkiness	**(R)** Object bulkiness. It describes how much the object bulges.
Area_convex	**(R)** Area of object's convex hull in pixels
Rect_dx	**(R)** Greater extent (width) of rotated min area enclosing rectangle
Rect_dy	**(R)** Smaller extent (height) of rotated min area enclosing rectangle
Rect_angle	**(A)** Rotation angle of minimum area rectangle [0,2PI]
Convexity	**(R)** Object convexity (how convex the object is)

Similar tools

“Get Objects” and “Get Object Features 2” tools.

Usually combined with

The “Get Object” tool, which creates input ODB buffer for this tool.

Tips & Tricks

Examples

Not indexed yet – sorry.

10.3. Get Object Features 2

Group

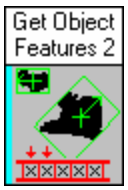
Object analysis tools.

Short description

The tool calculates selected features of all ODB objects or from an object list and stores feature values into the point-list buffer.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool calculates and extracts object features from the "object data base" (ODB) buffer with index **ODB_id**, created by previous call to the "**Get Objects**" tool. In case of invalid or missing ODB buffer the tool returns error 9118. The tool stores values of calculated features into the point-list buffer, starting from offset **Res_start**. The tool is able to get simultaneously up to 8 float features plus one point. If you need more features, you should call the tool several times with different result point-list sections and different feature request arguments.

***ATTENTION**. If the input ODB buffer has been created by "Get Objects" in **unconnected** mode, then you can get basic features only (see the description of the "Get Objects" tool and the section **Feature-calculation groups**) - mass center, parameters of the non-rotated rectangle, which encloses all foreground pixels (rectangle center, width, height and top/left corner), beginning contour point, area and average brightness.*

The ODB objects have indexes from 0 to **ODB_cnt** -1, where **ODB_cnt** is the number of ODB objects returned as result by the "**Get Objects**" tool. The tool selects indexes of objects to process in 3 modes, specified by the "**Obj_sel**" argument:

- **All** - all ODB objects.
- **List** - ODB objects are selected by a point-list section, specified by the arguments **List_start**, **List_size** and **List_param**. The list contains indexes of objects, the features of which we want to get.
- **One object** - selected by the **Obj_id** argument.

The tool works in two general modes, specified by the **Mode** argument:

- **Get features**. The tool calculates object features and stores features values into the result point-list section – see the description below.
- **Get contours**. The tool finds object contours and stores the contour points of the selected objects into the result point-list section – see the description below.

"Get features" mode

In this mode the arguments **Res_point** and **Res_param0** - **Res_param7** specify which features should be calculated and stored into respective point-list item fields:

Argument	Argument option	Description
Res_point	**None**	Don't store feature into item's point
	Mass center	Center of mass
	Rect center	Center of non-rotated enclosing rectangle
	Contour point	Beginning contour point from outer object contour
Res_param0-7	**None**	Don't store feature into respective parameter
	X_cent_r	X-coordinate of center of non-rotated enclosing rectangle
	Y_cent_r	Y-coordinate of center of non-rotated enclosing rectangle
	Width	Width of object's non-rotated enclosing rectangle
	Height	Height of object's non-rotated enclosing rectangle
	X_min	X-coordinate of top/left corner of non-rotated rectangle
	Y_min	Y-coordinate of top/left corner of non-rotated rectangle
	Area	Object area
	X_cent	X-coordinate of object's mass center
	Y_cent	Y-coordinate of object's mass center
	Avg_brightness	Average object brightness = sum(gray-level object pixels) / area
	Cont_len	Length of outer object's contour
	Hole_cnt	Number of object holes
	Min_dist	Min distance from the mass center to a contour point (outer contour)
	Max_dist	Max distance from the mass center to a contour point (outer contour)
	Compactness	Object compactness
	Circularity	Object circularity
	Major_rad	Greater radius of equivalent ellipse
	Minor_rad	Smaller radius of equivalent ellipse
	Ellip_angle	Rotation angle of equivalent ellipse [0,2PI]
	Anisometry	Object anisometry (elongatedness)
	Bulkiness	Object bulkiness. It describes how much the object bulges.
	Area_convex	Area of object's convex hull
	Rect_dx	Greater extent (width) of rotated min area enclosing rectangle
	Rect_dy	Smaller extent (height) of rotated min area enclosing rectangle
	Rect_angle	Rotation angle of minimum area rectangle [0,2PI]
	Convexity	Object convexity (how convex the object is)

Note that the point features "**Mass center**" and "**Rect center**" are available in two forms – as points and as separate X/Y coordinates.

Features of invalid objects

For each object with invalid index, the tool stores one point-list item with the following parameters:

- Point: x,y = -1,-1, subx,suby = -1,-1
- Param 0 to Param 7 = -1

"Get contours" mode

In this mode the arguments **Res_point** and **Res_param0** - **Res_param7** are ignored and the tool stores object contour points into the result point-list in a fixed format described below. The unused item fields are set to 0. The contour data is ordered according to the object sequence, defined by the **Obj_sel** argument and/or the "list" arguments. The next table shows contours of objects, selected for example by a list with indexes **`i0, i1, i2,…`**

Format of result point-list with contour data:

Object	Item	Item field	Description	Contour
i0 (first)	0	Point	First contour point of object's 1st contour	Contour 0
		Param0	Length of object's 1st contour (outer) = **N**	
		Param1	Number of contours for this object = (hole_cnt+1)	
		Param2	Total length of contour data for this object (# of point-list items). Adding Param2 to **Res_start** will jump to the start address of the next object contour data.	
		Param3	Total number of objects in the result point-list	
	1	Point	Next contour point	
	2	Point	Next contour point	
	. . .	. . .	. . .	
	N -1	Point	Last point of first object contour	
	0	Point	First contour point of next contour (if any)	Contour 1
		Param0	Length of next contour = **N1**	
	1	Point	Next contour point	
	2	Point	Next contour point	
	. . .	. . .	. . .	
	N1 -1	Point	Last point of next contour	
	. . .	. . .	. . . (next contours of current object)	Contour 2
i1	0	Point	First contour point of object's 1st contour	Contour 0
		Param0	Length of object's 1st contour (outer) = **N**	
		Param1	Number of contours for this object = (hole_cnt+1)	
		Param2	Total length of contour data for this object (# of point-list items). Adding Param2 to current point-list address will jump to the start address of the next object contour data.	
	1	Point	Next contour point	
	2	Point	Next contour point	
	. . .	. . .	. . .	
	N -1	Point	Last point of first contour	
	0	Point	First contour point of next contour (if any)	Contour 1
		Param0	Length of next contour = **N1**	

	1	Point	Next contour point	
	2	Point	Next contour point	
	. . .	. . .	. . .	
	N1 -1	Point	Last point of next contour	
	. . .	. . .	. . . (next contours of current object)	Contour 2
i2	. . .	. . .	. . . (next object contours)	Contour 0

In "**Get contours**" mode the actual number of stored point-list items is returned in the **Res_size** result. **Param3** of the 1st result item holds the actual number of stored objects. If the result list is not overflowed (enough **Max_res_size** for all contour data) then:

- **Param3** = result **Obj_cnt**
- **Res_size** <= **Max_res_size**

In case of result point-list overflow the tool returns the following results:

- **Param3** < **Obj_cnt**.
- **Res_size** = **Max_res_size** + 1

Contour data of invalid objects

For each object with invalid index, the tool stores in the current point-list location one item with zero parameters:

- Param0 = length of 1st contour = 0
- Param1 = # of contours for this object = 0
- Param2 = total length of contour data for this object = 0

Feature-calculation groups

If a requested feature is not calculated yet, then the tool calculates it. The object features are divided in 4 groups in respect to the type of the applied calculations. **Remember that when you request a feature from a given group, all features from that group will be calculated simultaneously**.

Group 1 - Basic features, calculated by "Get objects":	
Mass center	Center of mass
Rect center	Center of non-rotated enclosing rectangle
Contour point	Beginning contour point (a point from the outer object's contour)
Width	Width of object's non-rotated enclosing rectangle
Height	Height of object's non-rotated enclosing rectangle
X_min	X-coordinate of top/left corner of non-rotated enclosing rectangle
Y_min	Y-coordinate of top/left corner of non-rotated enclosing rectangle
Area	Object area in pixels
Avg_brightness	Average object brightness = sum(gray-level object pixels) / area
Group 2 - Contour features:	
Cont_len	Length of outer object's contour
Hole_cnt	Number of object holes
Min_dist	Min distance from the mass center to a contour point
Max_dist	Max distance from the mass center to a contour point
Compactness	Object compactness
Circularity	Object circularity

-	Object's contour data
Group 3 - Equivalent ellipse features:	
Major_rad	Greater radius of equivalent ellipse
Minor_rad	Smaller radius of equivalent ellipse
Ellip_angle	Rotation angle of equivalent ellipse [0,2PI]
Anisometry	Object anisometry (elongatedness)
Bulkiness	Object bulkiness. It describes how much the object bulges.
Group 4 - Min area rectangle features:	
Area_convex	Area of object's convex hull
Rect_dx	Greater extent (width) of rotated min area enclosing rectangle
Rect_dy	Smaller extent (height) of rotated min area enclosing rectangle
Rect_angle	Rotation angle of minimum area rectangle [0,2PI]
Convexity	Object convexity (how convex the object is)

Forced calculations

Some features need features from other groups. The features, which force calculations of other groups, are described below:

- **Group 1 - Basic features.** These features are calculated and stored into ODB by the "Get objects" tool. The features from this group do not force calculation of other groups.
- **Group 2 - Contour features.** These features need valid input ODB buffer, created by "Get objects". The features from this group do not force calculation of other groups.
- **Group 3 - Ellipse features.** These features need valid input ODB buffer, created by "Get objects". The features from this group do not force calculation of other groups with one exception - the feature **Ellip_angle** forces calculation of group 2.
- **Group 4 - Min area rectangle features.** These features need valid input ODB buffer, created by "Get objects". The features from this group force the calculation of group 2.

When an uncalculated feature is requested, the tool calculates and stores all features from its group into ODB. Next calls to the tool with requests for features from this group (for the same object from the same ODB buffer) do not re-calculate features and just read the ready feature values.

The most calculation-intensive group is Group 4 (see the section **Typical execution times**). If you need for example object orientation angles, you can improve the execution speed by using the ellipse orientation angles, so you may skip the calculations of the min area rectangles. This is possible if the processed objects are well described by equivalent ellipses.

Orientation angles

The object orientation angles are VIMOS angles in units PI/1000. Angle 0 is at 12 o'clock (vertical up) and the angles increase clockwise. Note that some tools show the VIMOS angles in degrees - for example if you link an angle result to a text-box, you will see the angle in degrees.

The tool calculates two types of orientation angles:

- equivalent ellipse angle **Ellip_angle**
- min area rectangle angle **Rect_angle**

Remember that each of these angles is not perfect in all cases. Some objects are described well by equivalent ellipses while others – by rotated min area enclosing rectangles.

***ATTENTION**. If the contour does not contain enough orientation data (1-pixel object for example), then the ellipse angle is in the range [0,PI] and the tool returns with error BLOB_R_NO_ORIENT (9116). In case of valid contour data the ellipse orientation angle is in the range [0,2PI].*

If the contour does not contain enough orientation data (1-pixel object for example), then

*the rectangle angle **Rect_angle** is in the range [0,PI] and the tool returns error BLOB_R_NO_ORIENT (9116). In case of valid contour data the rectangle orientation angle is in the range [0,2PI].*

An object is well oriented (i.e. it has good orientation data) if the point from the outer object contour, which is located at the maximum distance from the mass center, is single and distinct. Single-pixel objects have no orientation.

Shape features

The shape features provide valuable information about object shape because shape classification is an important task in the machine vision applications. The features **Compactness**, **Circularity**, **Anisometry** and **Bulkiness** show how similar an object is to a circle. The feature **Convexity** shows how convex an object is. Remember that some theoretical feature values (1's for perfect circles) sometimes can't be reached due to the digital nature of the applied calculations.

Compactness

Formula:	`Compactness = (Cont_len * Cont_len) / (4 * PI * Area)`

This feature describes how compact an object is compared to the perfect circle:

1: for perfect circle

>1: for other shapes.

As it is seen from the formula, the compactness grows quadratically with the length of the contour. Objects with holes have greater compactness values because their area decreases.

Circularity

Formula:	`Circularity = Area /(Max_dist * Max_dist * PI)`

This feature describes how circular an object is compared to the perfect circle:

1: for perfect circle

<1: for other shapes (not perfect circle and/or object with holes)

The circularity decreases for objects with holes because their area decreases.

Anisometry

Formula:	`Anisometry = Major_rad / Minor_rad`

This feature describes how elongated an object is compared to the perfect circle:

1: for perfect circle

>1: for other shapes

The anisometry increases for more elongated objects. If the feature is not available (degenerate objects may have `Minor_rad = 0`), it receives value 0.

Bulkiness

Formula:	`Bulkiness = (PI * Major_rad * Minor_rad) / Area`

This feature describes how much an object bulges compared to the perfect circle or ellipse:

1: for perfect circle or ellipse

>1: for other shapes

The bulkiness increases for objects with holes because their area decreases. It also increases for objects with more irregular contour forms. For objects with unavailable equivalent ellipse (1-pixel objects for example) the feature receives value 0.

Convexity

Formula:	`Convexity = Area / Area_convex`

This feature describes how much an object differs from its convex hull:

1: for perfect convex shape – circle or polygon
<1: for concave shapes or objects with holes

The convexity decreases for objects with holes and concavities. Non-calculated convexity features have zero values.

Degenerate objects

Objects for which some features can't be calculated (due to division by zero or other reason) are called ***degenerate*** objects. Such objects are for example the single-pixel objects and the objects composed of 1 pixel line. These objects have one or two zero ellipse/rectangle dimensions and therefore some features remain undefined. For example single-pixel objects have **Max_dist** = 0 and therefore the feature **Circularity** is not available due to division by zero.

Features of degenerate objects, which can't be calculated, receive undefined (not available) values, as described in the next section.

Values of uncalculated features

Features which are not calculated yet or can't be calculated because belong to degenerate objects receive the so called "not available" (**NA**) values:

`Cont_len = 0`
`Hole_cnt = -1`
`Min_dist = -1`
`Max_dist = -1`
`Compactness = 0`
`Circularity = 0`
	Contour features

`Major_rad = -1`
`Minor_rad = -1`
`Ellip_angle = -1`
`Anisometry = 0`
`Bulkiness = 0`
	Ellipse features

`Area_convex = 0`
`Rect_dx = -1`
`Rect_dy = -1`
`Rect_angle = -1`
`Convexity = 0`
	Rectangle features

Typical execution times

The pure execution time of the tool (without system overhead) depends on the size of the input image and the number of the processed objects. Here we present the execution times for a typical, not very complex image:

- Image size = 640 x 480
- Number of objects = 15
- Total number of object pixels = about 20000
- Min object area = 50

The execution times in milliseconds are:

Stage	Time (ms)
ODB generation (done by "Get Objects")	10
Contour detection and calculation of contour features	33
Calculation of ellipse features	14
Calculation of rectangle features	120

Tool drawing

The tool draws some features with adequate graphical representations, which are selected by the **Disp_mode** argument. Uncalculated features are not drawn. There is no sense to repeat drawing of features by several tools, which work with the same ODB buffer - choose only one of them, for

example the last. Note that the tool may draw nothing on error because features are not available at all (if the tool receives invalid or missing ODB buffer for example).

Object analysis errors

Code	Macro	Description
9100	BLOB_R_NO_MEMORY	Memory allocation error
9101	BLOB_R_IMG_OVF	Image buffer overflow
9102	BLOB_R_RLCBUF_OVF	RLC buffer overflow
9103	BLOB_R_BINIMG_ERR	Invalid format of binary image
9104	BLOB_R_MBUF_OVF	Object merge buffer overflow
9105	BLOB_R_OBJLABEL_ERR	Object labeling error
9106	BLOB_R_ODB_OVF	Object database (ODB) buffer overflow
9107	BLOB_R_INTERNAL_ERR	Internal program error
9108	BLOB_R_INV_OBJ_IDX	Invalid object index
9109	BLOB_R_TRACE_ERR	Contour tracing error
9110	BLOB_R_CONTBUF_OVF	Contour buffer overflow
9111	BLOB_R_INV_FEATURE	Invalid object feature
9112	BLOB_R_DELETED_OBJ	Deleted object
9113	BLOB_R_FEATURE_NOCALC	Feature not calculated
9114	BLOB_R_INVALID_ARG	Invalid input argument
9115	BLOB_R_DRAW_ERR	Feature drawing error
9116	BLOB_R_NO_ORIENT	Insufficient orientation data
9117	BLOB_R_COPY_PROT	Copy protection error
9118	BLOB_R_INVALID_ODB	Invalid ODB buffer
9119	BLOB_R_CONTOUR_LIMIT	Contour limit exceeded

Results:

The tool returns 6 float results – ODB identifier, number of objects in ODB, actual ODB size, total allocated ODB memory in bytes, actual size of result point-list (# of stored items) and total number of objects selected for processing. The tool stores also feature values or contour data into the point-list buffer, starting from **Res_start**.

The result **Obj_cnt** shows the total number of objects selected for processing. It is equal to:

- the result **ODB_cnt** - if all ODB objects are selected;
- the argument **List_size** - if an object list is used.

In **Mode** = "**Get features**" the result **Res_size** shows the actual number of processed objects:

- **Res_size == Obj_cnt** - no overflow;
- **Res_size < Obj_cnt** - result point-list overflow (result **Res_size** = argument **Max_res_size**).

In **Mode** = "**Get contours**" the result **Res_size** is the actual number of stored contour points in the result point-list:

- **Res_size** <= **Max_res_size** - no overflow;
- **Res_size** > **Max_res_size** - result point-list overflow, **Max_res_size** = # of stored items.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
ODB_id	ODB identifier [0,1023]
List_start	Start address of input point-list section with object list containing indexes of objects to get features from.
List_size	Size of object list section (# of items)
List_param	Point-list parameter Param0 - Param7, which holds the object indexes.
Obj_id	Object index, used to select 1 ODB object
Obj_sel	Object selection mode: • **All**. Process all ODB objects. • **List**. Process objects selected by **List_start**, **List_size** and **List_param**, which specify indexes of objects of interest. • **One object**. Process object with index **Obj_id**.
Mode	Tool operation mode: • **Get features**. Calculate and read object features specified by **Res_point** and **Res_param0-7** into the result point-list section. • **Get contours**. Read object contours into the result point-list section.
Cont_limit	Contour limit used in the calculation of the contour features. Specifies max number of detected contour points in one contour before the tool aborts execution with error 9119. **0 = don't check**
Res_start	Start address of output point-list section where the tool stores features values or contour data.
Max_res_size	Max size of output point-list section (# of items).
Res_point	Specifies which point feature should be calculated and stored into the point field of the result point-list items: • **None**. Don't calculate and store point feature. • **Mass center**. Center of mass. • **Rect center**. Center of non-rotated enclosing rectangle. • **Contour point**. Beginning contour point.
Res_param0	Specifies a feature to calculate and store into parameter 0 of the result point-list items – see tool description, "**Get features**" mode.
Res_param1	Specifies a feature to calculate and store into parameter 1 of the result point-list items – see tool description, "**Get features**" mode.
Res_param2	Specifies a feature to calculate and store into parameter 2 of the result point-list items – see tool description, "**Get features**" mode.
Res_param3	Specifies a feature to calculate and store into parameter 3 of the result point-list items – see tool description, "**Get features**" mode.
Res_param4	Specifies a feature to calculate and store into parameter 4 of the result point-list items – see tool description, "**Get features**" mode.
Res_param5	Specifies a feature to calculate and store into parameter 5 of the result point-list items – see tool description, "**Get features**" mode.
Res_param6	Specifies a feature to calculate and store into parameter 6 of the result point-list items – see tool description, "**Get features**" mode.
Res_param7	Specifies a feature to calculate and store into parameter 7 of the result point-list items – see tool description, "**Get features**" mode.
Draw_clr	Tool drawing color on success

Err_clr	Tool drawing color on error
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only
Disp_mode	Feature display mode. Add the number 1,2,4,… to enable the display of the respective feature: • bit 0 : draw object numbers (add 1) • bit 1 : draw cross markers of mass centers (add 2) • bit 2 : draw non-rotated enclosing rectangles (add 4) • bit 3 : draw contours (add 8) • bit 4 : draw equivalent ellipses (add 16) • bit 5 : draw min area rotated rectangles (add 32)

Results

Result	Description
ODB_id	ODB identifier, equal to the argument ODB_id.
ODB_cnt	Number of objects in ODB.
ODB_size	Actual ODB size in bytes excluding the free space.
ODB_tot_size	Total allocated ODB memory in bytes.
Res_size	Actual # of items stored in the result point-list section (**Res_size** <= argument **Max_res_size**).
Obj_cnt	Total number of objects, selected for processing – see “Results” in the **Description** section.

Similar tools

“Get Objects” and “Get Object Features” tools.

Usually combined with

The “Get Object” tool, which creates input ODB buffer for this tool.

Tips & Tricks

Examples

Not indexed yet – sorry.

11. Other tools

The tools from this group have miscellaneous functions, which do not fall into one of the basic tool groups. Here are tools for image taking, flash file handling, timers, calibration, etc.

Editor toolbar of this group of tools:

11.1. Take Image

Group

Other tools.

Short description

The tool gets an image from the CCD sensor and stores the image into the frame buffer (the data source for image-processing tools).

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons

Description

General function:

The tool takes an image from the CCD sensor and saves the image into the frame buffer of the VIMOS kernel. Tool arguments control the image taking operation. The tool has no results. Remember that VIMOS kernel supports automatic image taking in each user-program cycle in run-modes "Live & Shoot" and "Shoot & Show". Disable automatic image taking when using the tool (recommended).

The **Shutter** argument specifies exposure time in microseconds.

The **Video mode** argument sets camera video mode, which is set after the image-taking operation:

- **Freeze** (default) – the camera goes into "still" video mode and the taken image is shown on the monitor as still frame.
- **Live** – the camera goes into "live" video mode, the camera takes continuously images and shows them on the monitor.

The **Wait ext. trigger** argument specifies triggered image taking when the tool waits for an external trigger before taking the image:

Wait ext. trigger	Description
No (default)	Unconditional Image taking without external trigger.
IN0=1	The tool waits for PLC input IN0 set to 1.
IN0: 0->1	The tool waits for change of PLC input IN0 from 0 to 1.
TRIGGER	The tool waits for external trigger interrupt and takes image using the VCRT functions tenable() and trdy().

The **Parallel proc** argument specifies parallel image taking and user-program execution (currently supported on TI cameras only):

- **No** (default) – parallel operation is disabled.
- **Yes** – parallel operation is enabled. The tool waits for the completion of the image taking operation started in the previous user-program cycle, starts a new image taking operation and returns. The image processing tools work with the image taken by the tool in the previous cycle. The new image will be processed in the next cycle. This option can be combined with triggered image taking – the tool waits for trigger before starting the image taking.

***ATTENTION**. Notes on parallel image taking.*

The aim of the parallel take image processing is to increase the system performance on the TI camera. For example a typical take image operation with 10ms exposure time (the shutter value) will take about 33-34 ms. Suppose your image processing tools work for 50 ms. In non-parallel mode one system pass will take 33 + 50 = 83 ms. In parallel mode the take image time is hidden in the image processing time and each system pass will take 50 ms. Note than you can't receive execution time less than the time of the image taking tool (plus some system overhead time).

Remember that you will miss the first picture when you start user-program in parallel mode. The first system pass will use the old picture in the frame buffer, which may be indeterminate. If you want to process correct first image, you should take image in cycle 0 in non-parallel mode.

When you stop program execution, the last image taken by the tool is stored in the frame buffer, but is not processed.

In triggered image taking, the results of one system pass refer to the image, taken in the previous pass by the previous trigger event.

In case of missing trigger event the system will hang. You can break the endless wait loop by setting a break trigger, which is enabled by the **Break wait** argument. The "**PLC**" option selects as break trigger the PLC input **IN3** (**IN1** on sensor cameras M40, M50), set to 1. The "**Serial**" option breaks the wait loop on any received data via the serial port or the keypad port (TI only) of the camera. You may combine both options.

ATTENTION. Notes on triggered image taking:

The TRIGGER option is valid for progressive scan cameras only (VC38, VC2038). It should be used in camera video-modes 1,3,5,7 ("Shoot & show" or "Show"). The tool attempts to read an image by the fastest possible method depending on the camera model.

*To achieve best performance in non-TRIGGER mode, use the tool in **Live** run-mode on ADSP camera and **Show** run-mode on TI camera. This will disable the automatic image taking in each system cycle.*

ATTENTION. On FA45 the tool takes a color image in Bayer matrix format. Use the "Take CMOS Image" tool to take a gray-level image on FA45.

Results:

The tool returns 1 float result – error code of the image taking operation (0=OK).

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Shutter(uS)	Shutter value in microseconds.
Video mode	Video mode set after image taking.
Wait ext. trigger	Wait for external trigger IN0 or TRIGGER before image taking
Parallel proc	Parallel image taking and user-program execution.
Break wait	Trigger event, which breaks the “external trigger” wait loop – none, PLC input, serial data or both PLC and serial data.

Results

Result	Description
Result	Image taking error code: 0 - success 5030 – trigger break event occurred, image is taken without trigger 5031 – tenable() error 5032 – tpstart() error in parallel image taking

Similar tools

Take CMOS image.

Usually combined with

Image processing tools, which work with the image, taken by the tool.

Tips & Tricks

Examples

Not indexed yet – sorry.

11.2. Take CMOS Image

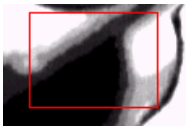
Group

Other tools.

Short description

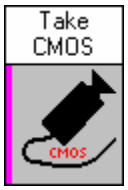
This tool takes image on ADSP-based sensor cameras (VCM40 (color), VCM50) and TI-based FA45.

VIMOS kernel presentation



Note: This is the tool's presentation in the Simulator. On sensor-cameras (no video-output, no overlay) the tool has no graphical representation.

Editor icons



Description

The tool takes an image from the camera sensor and stores the image in the frame buffer. VCM40 supports getting part of the sensor area, which increases the tool's speed. Dedicated tool arguments specify the screen rectangle of interest. On VCM40 the tool can take a gray-level image, a multi-color image or a single-color image with size ¼ of the specified rectangle. On VCM50 the tool takes a full-size, ½ or ¼ gray-level image (binning mode). On FA45 the tool takes a full-size color image or a gray-level image.

ATTENTION. On VCM40 the tool calls the VCRT function setfield(). This function allows setting a rectangular region for subsequent readout from the CMOS sensor. The parameters (x0, y0) define the left upper corner of the region. dx and dy determine the horizontal and vertical size of the window. All parameters must be multiples of 4, otherwise the values will be truncated. The function automatically adjusts for dark pixels surrounding the active matrix. So, values for x0 and y0 start with 0 for the first active row and column available on the sensor.

Color images are coded in the so called Bayer matrix format. Each color pixel is coded in the output image by a 2x2-pixel matrix in two sequential image rows and columns:

	1st column	**2nd column**
1st row	**G1** (green 1)	**R** (red)
2nd row	**B** (blue)	**G2** (green 2)

The actual RGB components of each color pixel are R, (G1 + G2)/2 and B.

The color image on **VCM40** is coded in the following Bayer matrix format (top/left corner = G):

```
G R G R G R G . . .
B G B G B G B . . .
G R G R G R G . . .
B G B G B G B . . .
. . . . . . . . . .
```

The color image on **FA45** is coded in the following Bayer matrix format (top/left corner = R):

```
R G R G R G R . . .
G B G B G B G . . .
R G R G R G R . . .
G B G B G B G . . .
. . . . . . . . . .
```

The **Wait ext. trigger** argument specifies triggered image taking when the tool waits for an external trigger before taking the image:

Wait ext. trigger	Description
No (default)	Unconditional image taking without an external trigger.
IN0=1	The tool waits for PLC input IN0 set to 1.
IN0: 0->1	The tool waits for change of PLC input IN0 from 0 to 1.
TRIGGER	N/A on sensor cameras and FA45, same as "**No**" option

In case of missing trigger event the system will hang. You can break the endless wait loop by setting a break trigger, which is enabled by the **Break wait** argument. The "**PLC**" option selects as break trigger the PLC input **IN1** (sensor cameras have 2 PLC inputs IN0 and IN1). The "**Serial**" option breaks the wait loop on any received data via the serial port of the camera. You may combine both options. The M40/M50 camera button (continuous press) also breaks the wait loop and enters the VIMOS shell.

All arguments except the color ones are valid on VCM40. The following arguments are valid on VCM50 and FA45:

- **Illumination**
- **Shutter (uS)**
- **Mode**

  On FA45 the first two options of the argument specify the type of the taken image – **B&W** takes a gray-level image and **Color** takes a color image in a Bayer matrix format. The **B&W** mode is slower than the color mode. The remaining options of this argument are ignored.

  On VCM50 the two options **Binning 1** and **Binning 2** take ½ or ¼ of the screen image in binning mode. Note that these options increase about two times the image brightness. All other options are ignored.
- **Wait ext. trigger**
- **Break wait**

The **Red** gain argument is used to set gain on M50. **Remember that on M50** this argument value is in relative units in the range [1,19] – a value of 1 corresponds to actual gain of 14 (the default gain value 10 actually sets gain of 140).

The last three color arguments are used in the Simulator only, where the image rectangle is drawn on the screen.

Results:

The tool returns 1 float result – error code of the image taking operation (0=OK).

Arguments

Argument	Description
Point	Center point of screen rectangle, which specifies sensor area to take image from and frame buffer area, where sensor data is stored. **Used on VCM40 only.**
Width	Width of image rectangle (should be multiple of 4). **Used on VCM40 only.**
Height	Height of image rectangle (should be multiple of 4). **Used on VCM40 only.**
Red	CMOS sensor gain: RED on VCM40 in the range [0,255]. Gain on VCM50 in relative units in the range [0,19]. Argument value of 10 corresponds to the default VCM50 gain of 140. **Used on VCM40 and VCM50.**
Green 1	CMOS sensor gain: GREEN 1. **Used on VCM40 only.**
Green 2	CMOS sensor gain: GREEN 2. **Used on VCM40 only.**
Blue	CMOS sensor gain: BLUE. **Used on VCM40 only.**
Illumination	Set illumination during image-taking process: • **Illum&Mark off**. Illumination and mark projector off. • **Mark proj. on**. Mark projector on. • **Illum. on**. Illumination on.
Shutter (uS)	Shutter value in microseconds.
Video mode	Specifies video mode after image taking. **Currently ignored.**
Wait ext. trigger	Wait for external trigger IN0 before image taking.
Mode	Image-taking mode: • **B&W**. Take a gray-level image. • **Color**. Take a multi-color image in Bayer matrix format. • **R**. Take a single-color R image (1/4 of image size). • **G1**. Take a single-color G1 image (1/4 of image size). • **G2**. Take a single-color G2 image (1/4 of image size). • **B**. Take a single-color B image (1/4 of image size). • **Binning 1**. Take ½ image (640 x 240) in binning mode on VCM50. This option increases approximately twice the image brightness – use normal **Shutter** / 2. • **Binning 2**. Take ¼ image (640 x 120) in binning mode on VCM50. This option increases approximately 4 times the image brightness – use normal **Shutter** / 4.
Break wait	Trigger event, which breaks the “external trigger” wait loop – none, PLC input, serial data or both PLC and serial data.

Draw_clr	Tool drawing color on success (0=default : green). **Used in Simulator only.**
Err_clr	Tool drawing color on error (0=default : red) **Used in Simulator only.**
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only **Used in Simulator only.**

Results

Result	Description
Result	Image taking error code: 0 - success 5030 – trigger break event occurred, image is taken without trigger

Similar tools

Take image.

Usually combined with

Color pixel counter tool.

Tips & Tricks

Examples

Not indexed yet – sorry.

11.3. Copy Image

Group

Other tools.

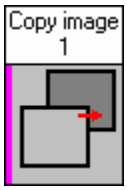
Short description

The tool copies image between frame buffer and freeze buffer.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool transfers full-screen image between frame buffer and freeze buffer. All image processing tools work with the frame buffer. The freeze buffer serves as a spare image buffer, where you can save current frame buffer and restore it later. This tool works only in **run-mode**.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Copy direction	Specifies source and target image buffer: • **Frame >> Freeze** - copy contents of frame buffer into freeze buffer. • **Frame << Freeze** - copy contents of freeze buffer into frame buffer.

Results

The tool has no results.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

11.4. Button & LEDs

Group

Other tools.

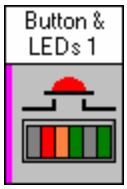
Short description

This tool sets 6 LEDs and returns the state of the VCM40/VCM50 button. On TI camera without video-output like FA45 the tool sets the 4 LEDs on the back panel (a button is not present).

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

Sensor camera (VCM40/VCM50)

The tool displays the 6-bit value of the argument "LED value" on the camera LEDs. The 6-bit mask argument "LED mask" controls setting of the LED bits. Unmasked LED bits (mask bit = 0) are set from the corresponding "LED value" bits. Masked LED bits (mask bit = 1) are kept unchanged. LED mask equal to 63 (0x3F) keeps all LED bits unchanged. The LED bits can be set in three different colors – green, red and amber.

The second function of the tool is to check the state of the built-in camera button and to return the button state (non-zero value if button is pressed). Since the button is also used to enter M40/M50 camera shell, the following algorithm is observed when working with shell and "Button & LEDs" tool:

- By default the button-press event (continuos pressing, because button input is not buffered by interrupt) is always used to enter shell. This is the case when there is no "Button & LEDs" tool in the user program. When you put such tool in the program, you should specify the target, which processes the button event. "Enable shell" = No specifies that button events are processed by the tool and shell is disabled. Set "Enable shell" = Yes if you want to preserve access to shell. The argument option "No change" keeps current setting of enabled/disabled shell.
- If you have disabled the shell, the only way to enter the shell is to execute program branch with "Button & LEDs" tool, which enables shell. This could be done for example as response to given values, read from the PLC inputs (see "Get I/O value" tool). Note that sensor cameras like VCM40 and VCM50 have two PLC inputs only – IN0 and IN1.
- You may enter shell directly from the tool by "Enter shell" = Yes. Remember that each time you execute tool with this option, you will break execution of the user program. Restart program by the "r" (run) shell command. Below is given a brief description of shell commands, which replace the "edit" mode of VIMOS kernel on cameras without video output.

***INFORMATION**. Information about the VIMOS shell on sensor cameras can be found in the manual "Using VIMOS on sensor camera".*

TI camera without video-output (FA45)

The tool displays the 4-bit value of the argument "LED value" on the camera LEDs. A 4-bit mask argument "LED mask" controls setting of the LED bits. Unmasked LED bits (mask bit = 0) are set from the corresponding "LED value" bits. Masked LED bits (mask bit = 1) are kept unchanged. LED mask equal to 15 (0xF) keeps all LED bits unchanged. Argument values above 15 are truncated to 4 least-significant bits. The led colors are fixed and can't be set by the tool. The bit assignment is:

Bit	LED color	Description
bit 0	green	Leftmost LED : Power
bit 1	red	2nd LED : Error
bit 2	yellow	3rd LED: Q1
bit 3	yellow	Rightmost LED: Q2

The arguments "LED color", "Enable shell" and "Enter shell" are ignored. Camera button is not present and the tool returns always a zero result.

Results:

The tool returns 1 float result equal to 1 if the button is in pressed state (valid on VCM40/VCM50).

Arguments

Argument	Description
LED mask	6-bit mask for LED bits. Each bit in the mask specifies how to set respective LED bit from the "LED value" argument: 0 : set LED bit from respective "LED value" bit 1 : keep LED bit unchanged **4-bit mask on FA45.**
LED value	6-bit LED value 0-63. LED bits are set if corresponding mask bits are 0. **4-bit value on FA45.**
LED color	Color of set LED bits: green, red or amber. **Ignored on FA45.**
Enable shell	Enable shell (button press enters the shell): • **No**. Disable shell, button events are reserved for this tool. • **Yes**. Enable shell, button events are reserved for shell. • **No change**. Enabled/disable settings are not changed. **Ignored on FA45.**
Enter shell	Enter shell unconditionally. **Ignored on FA45.**

Results

Result	Description
Button state	State of M40/M50 button: 0=released, 1=pressed **Always 0 on FA45.**

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

11.5. Pause

Group

Other tools.

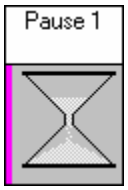
Short description

The tool delays user-program execution by given time interval or stops execution and waits for left mouse click to resume.

VIMOS kernel presentation

Note: This is the tool representation in “Wait click” mode. In “Wait time” mode the tool does not draw in the overlay picture.

Editor icons



Description

General function:

The tool delays or stops the execution of the user-program. The argument “Mode” specifies mode of tool operation. In “Wait time” mode the tool waits “Duration” milliseconds and then resumes execution. In “Wait click” mode the tool stop execution and waits for left mouse click to resume. This tool works in **run-mode** only.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Mode	Specifies mode of tool operation: • **Wait time.** Wait “Duration” ms and continue execution. • **Wait click**. Wait for left mouse click and continue execution (“Duration” is ignored).
Duration	Delay time in milliseconds.
Draw_clr	Tool drawing color on success (0=default : green).

	Currently not used.
Err_clr	Tool drawing color on error (0=default : red). **Currently not used.**
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only **Currently not used.**

Results

The tool has no results.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

11.6. Timer

Group

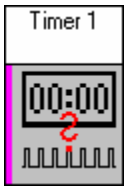
Other tools.

Short description

The tool measures time intervals in ms.

VIMOS kernel presentation

Editor icons



Description

General function:

The tool measures elapsed system time and displays measured time in the overlay picture. You may select up to 10 different timers. The tool works in two modes of operation:

- Reset mode – the timer is cleared in each execution cycle. In this mode the tool measures the time passed between two successive tool executions in the user-program loop.
- Accumulation mode – the timer shows the total time passed from the start of the user program.

The time is displayed in the following format:

`HH:MM:SS.NNN`

where:

`HH`	=	hours
`MM`	=	minutes
`SS`	=	seconds
`NNN`	=	milliseconds

The tool does not work in **edit-mode** of VIMOS kernel.

Results:

The tool returns one result – measured time in milliseconds.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Position	Point argument, which specifies position of timer text box in the screen.
Timer #	Number of the timer to use (0 - 9).
Reset	Sets mode of timer operation: **Yes** = reset mode, **No** = accumulate mode.
Text Size	Size of timer text box: 1 = normal, 2 = large.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Measured time	Elapsed system time in milliseconds.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

11.7. Delete Flash Files

Group

Other tools.

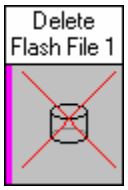
Short description

The tool deletes flash files and does fast flash-packing.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool deletes flash files and packs the flash. The names of the flash files must be stored in the string buffer as a sequence of 0-terminated strings. If the file count is 0, the tool performs packing operation only. The tool works on ADSP and TI cameras. The "File drive" argument is ignored on ADSP camera. On PC the tool deletes file(s) in current workspace folder, packing is not supported.

Results:

The tool returns 1 float result – size of remaining free flash space in Kbytes after packing.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
File name start	Start position of "File count" 0-terminated file names in the string buffer.
File count	Number of files to delete (0 – pack only).
File drive	Specifies file drive, which contains files to delete and which should be packed: • **Flash**. Flash EPROM on ADSP and TI cameras.

	• **MMC**. Multimedia card on TI camera (N/A on ADSP camera). Currently packing of MMC is not available.

Results

Result	Description
Free flash	Free flash space in Kbytes after pack operation.

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

11.8. Free Pattern

Group

Other tools.

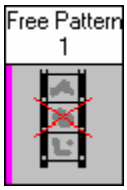
Short description

The tool deletes a pattern loaded by the “Correlation Init” tool.

VIMOS kernel presentation

This tool has no graphical representation. It doesn’t draw into the overlay picture.

Editor icons



Description

General function:

The “Correlation Init” tool creates and stores image patterns in reserved VIMOS memory buffer called the “FAT” buffer. Patterns are used by “Correlation Exec” tool to perform pattern matching of non-rotated and non-scaled image objects. Patterns remain in FAT buffer until removed by the “Free pattern” tool. If you execute the “Correlation Init” tool in a loop, you may soon overflow the FAT buffer, if you don’t free redundant patterns. This tool frees one or all patterns in the FAT buffer.

***ATTENTION**. The FAT buffer is cleared:*

- *On VIMOS kernel start.*
- *When a new user-program file is loaded into kernel.*
- *When you begin creation of a new user-program in edit mode.*

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Pattern num	Number of pattern to be freed if “Option” = Designated. Can be linked to the result of the respective “Correlation Init” tool, which have created the pattern.
Option	Specifies mode of tool operation: • **Designated**. Free the pattern with number “Pattern num”.

	• **All**. Free all patterns.

Results

The tool has no results.

Similar tools

Usually combined with

“Correlation Init” tool.

Tips & Tricks

If you need to learn an image pattern by “Correlation Init” and then search it multiple times by “Correltaion Exec”, we recommend executing once “Correlation Init” in cycle 0 of the user-program. Keep the pattern until your program executes “Correltaion Exec” in a loop. Free the pattern when you break the loop.

Examples

Not indexed yet – sorry.

11.9. Calibrate

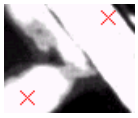
Group

Other tools.

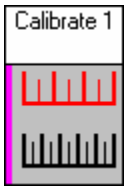
Short description

This tool generates and optionally activates calibration set.

VIMOS kernel presentation



Editor icons



Description

General function:

The tool generates calibration parameters, based on two input points and two float real-distance values. Two points with coordinates (x1,y1) and (x2,y2) define two screen distances in pixels:

```
screen_dx = abs(x1 – x2)      horizontal screen distance
screen_dy = abs(y1 – y2)      vertical screen distance
```

When executing tools, which measure distances, VIMOS kernel converts horizontal pixel distance `screen_dx` into real-world distance, equal to “X real distance”. Similarly, the kernel converts vertical pixel distance `screen_dy` into real-world distance “Y real distance”. Real distance can be measured in 5 different units – pixels, mm, cm, inches, feet. The tool works with screen sub-pixel accuracy of 1/1000.

The tool saves calibration parameters into data set called ***calibration set***. VIMOS kernel supports 21 calibration sets, numbered from 0 to 20. Calibration set 0 is reserved (1:1) and can’t be modified. All calibration sets (1:1 by default) are stored into system memory buffer. One of these sets is active, which means the tools use it to transform pixel coordinates. You may use this tool just to modify the calibration set, specified by “Calibration set”, without changing the active set, or you may choose also to make the modified set active by the activate options of the “Activate/save” argument. Another possibility to select active set gives the tool “Select Calibration Set”.

ATTENTION*. Before each pass of the user-program in the main system loop, VIMOS kernel sets default calibration 1:1 (actually activates the reserved calibration set 0). If you create and activate calibration set in cycle 0 (see option “Activate”), in next cycles the active calibration set will be changed. Insert a “Select Calibration Set” tool to solve the problem.*

You can use horizontal calibration for vertical one or vice versa by setting respective option in the “Use calibration” argument. Choose “save” options of the “Activate/save” argument to save all calibration sets into flash file ***clb1.vm***. Avoid using this option in endless user-program loop because you may overflow the flash memory of the camera.

Results:

The tool has 1 float result – error code of calibration operation (0=OK).

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Point 1	First calibration point.
Point 2	Second calibration point.
X real distance	Real distance on horizontal axis.
Y real distance	Real distance on vertical axis.
Measuring unit	Real distance measuring unit - pixels, mm, cm, inches and feet.
Use calibration	Use horizontal calibration for vertical or vice versa: • **No**. Use separate calibration on the two axes. • **Use X for Y**. Use X calibration also as Y calibration. • **Use Y for X**. Use Y calibration also as X calibration.
Calibration set	Number of calibration set: 1 to 20.
Activate/save	Activate current calibration set and/or save all calibration sets to file: • **No**. Do not activate current set; do not save calibration sets into file. • **Activate**. Activate current (created by the tool) set. • **Save**. Save all calibration sets into file. • **Activ. & save**. Activate current set and save calibration sets into file.
Draw_clr	Tool drawing color on success (0=default : green)
Err_clr	Tool drawing color on error (0=default : red)
Draw_mode	Tool drawing mode: • 0 = draw on (always) • 1 = draw off (never) • 2 = draw on success only • 3 = draw on error only

Results

Result	Description
Calibration error	Calibration error code: 0 = OK.

Similar tools

Select Calibration Set.

Usually combined with

Select Calibration Set.

Tips & Tricks

Examples

Not indexed yet – sorry.

11.10. Select Calibration Set

Group

Other tools.

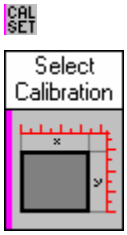
Short description

This tool activates a calibration set.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons



Description

General function:

The tool changes the active calibration set. The "**Set number**" argument specifies the number of the activated calibration set (from 1 to 20). Each cycle of the user-program execution begins with default calibration 1:1. The selected calibration set can be initialized (created) in two ways:

- In edit mode – interactive calibration using the "Calibrate" dialog box.
- In run mode – by the "Calibrate" tool.

Non-initialized sets use default calibration 1:1. All calibration sets are initialized from file ***clb1.vm*** on system start if the file is present.

Algorithm

Please read the description chapter above.

Arguments

Argument	Description
Set number	Specifies number of activated calibration set (1 to 20)

Results

The tool has no results.

Similar tools

Calibrate tool.

Usually combined with

Calibrate tool.

Tips & Tricks

Examples

Not indexed yet – sorry.

11.11. Calibrate Touch-Screen

Group

Other tools.

Short description

This tool calibrates the touch-screen.

VIMOS kernel presentation

This tool has no graphical representation. It doesn't draw into the overlay picture.

Editor icons

Description

General function:

The tool temporarily stops execution of the user-program and enters calibration of the touch-screen. Press two touch-screen spots, marked by small crosses in the upper left and the lower right screen corners. The tool saves touch-screen calibration settings in the system initialization file and resumes execution. Right mouse click aborts calibration without changing the settings.

The tool works on TI cameras only. The tool works in run-mode only.

***ATTENTION**. Be careful when putting this tool in the user-program because it stops execution and waits for user input from the touch-screen device. The program is not able to detect automatically presence or absence of a touch-screen device.*

Note:

VIMOS kernel on TI camera supports operation with touch-screen device, based on the AHL-51S "Analog Touch Panel Controller". The touch-screen device must perform serial communication at TTL levels (+/-5 V) and should be connected to the keypad port of the TI camera.

Results:

The tool has 1 float result – error code of the touch-screen calibration operation (0=OK).

Algorithm

Please read the description chapter above.

Arguments

The tool has no arguments.

Results

Result	Description
Calibration error	Touch-screen calibration error code (0 = OK).

Similar tools

Usually combined with

Tips & Tricks

Examples

Not indexed yet – sorry.

12. Pseudo-tools

The Editor supports several pseudo-tools, which actually don't have respective tools in VIMOS kernel. These tools support some system functions or generate multiple normal tools in VIMOS kernel.

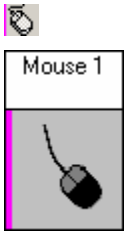
Editor toolbar of this group of tools:

There is no separate toolbar for this group of tools. Depending on their functions they are put into other tool groups:

- "Mouse" pseudo-tool – in the "Other" group.
- "Counters" pseudo-tool – in the "Statistics" group.
- "Calculator 2" pseudo-tool – in the "Graphics & Calculations" group.

12.1. Mouse

Editor icons



There is no Mouse tool in the VIMOS kernel. The Mouse pseudo-tool in Editor allows linking of point arguments to mouse, as it is possible in VIMOS kernel by the linkage mechanism. This is actually steering to the mouse, because the point argument retains its relative position on the screen. Note that in case of direct link all points linked to the mouse would snap to the same point on the screen.

***ATTENTION**. You should not link the Mouse's result directly to any tool's argument. Instead, you should link it to the corresponding steering arguments (`.steer.x` and `.steer.y`).*

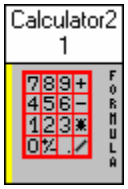
12.2. Counters

Editor icons

Place this tool at the beginning of the diagram if you are going to use counters in your program. The tool retrieves the values of several statistics counters. Each argument value specifies number of counter, the value of which is returned by the respective result. Currently there are 1023 counters numbered from 1 to 1023.

12.3. Calculator 2

Editor icons



This pseudo-tool generates a sequence of ordinary calculator tools in the user-program, which implement more complex calculation formula. The calculations are organized in a tree of binary or unary operations.

Build desired diagram with all necessary connections between tools. First you should link to the calculator 2 tool all tool results, which participate in the calculations. You can do this by opening tool context menus by right clicks and then linking tool outputs to calculator 2.

Now you are ready to program the calculator 2 itself. Open the “Properties” dialog by left double click. Select a root calculator operation by the ***Operator*** combo-box. Select left and right operands (left operand only for unary operations). Each operand can be a constant, a link to one of the selected already results or a formula, which adds next entry to the tree. Click on ***Constant*** (if enabled), ***Link*** or ***Formula*** radio-button to specify current left and right operand. Select a desired tool result by the ***Link*** combo-box. Selecting ***Formula*** will open a child window, which defines next level in the operation tree in a similar manner. The calculations are done as if operands and operation from a given tree level are closed in brackets.

APPENDIX A. ASCII table

Dec	Hex	Character	Dec	Hex	Character
000	00	NUL (Null char.)	032	20	(Space)
001	01	SOH (Start of Header)	033	21	! (exclamation mark)
002	02	STX (Start of Text)	034	22	" (double quote)
003	03	ETX (End of Text)	035	23	# (number sign)
004	04	EOT (End of Transmission)	036	24	$ (dollar sign)
005	05	ENQ (Enquiry)	037	25	% (percent)
006	06	ACK (Acknowledgment)	038	26	& (ampersand)
007	07	BEL (Bell)	039	27	' (single quote)
008	08	BS (Backspace)	040	28	((left/opening parenthesis)
009	09	HT (Horizontal Tab)	041	29	) (right/closing parenthesis)
010	0A	LF (Line Feed)	042	2A	* (asterisk)
011	0B	VT (Vertical Tab)	043	2B	+ (plus)
012	0C	FF (Form Feed)	044	2C	, (comma)
013	0D	CR (Carriage Return)	045	2D	- (minus or dash)
014	0E	SO (Shift Out)	046	2E	. (dot)
015	0F	SI (Shift In)	047	2F	/ (forward slash)
016	10	DLE (Data Link Escape)	048	30	0
017	11	DC1 (XON) (Device Control 1)	049	31	1
018	12	DC2 (Device Control 2)	050	32	2
019	13	DC3 (XOFF)(Device Control 3)	051	33	3
020	14	DC4 (Device Control 4)	052	34	4
021	15	NAK (Negative Acknowledgement)	053	35	5
022	16	SYN (Synchronous Idle)	054	36	6
023	17	ETB (End of Trans. Block)	055	37	7
024	18	CAN (Cancel)	056	38	8
025	19	EM (End of Medium)	057	39	9
026	1A	SUB (Substitute)	058	3A	: (colon)
027	1B	ESC (Escape)	059	3B	; (semi-colon)
028	1C	FS (File Separator)	060	3C	< (less than)
029	1D	GS (Group Separator)	061	3D	= (equal sign)
030	1E	RS (Request to Send)	062	3E	> (greater than)
031	1F	US (Unit Separator)	063	3F	? (question mark)

Dec	Hex	Character	Dec	Hex	Character
064	40	@ (AT symbol)	096	60	`
065	41	A	097	61	a
066	42	B	098	62	b
067	43	C	099	63	c
068	44	D	100	64	d
069	45	E	101	65	e
070	46	F	102	66	f
071	47	G	103	67	G
072	48	H	104	68	H
073	49	I	105	69	I
074	4A	J	106	6A	J
075	4B	K	107	6B	K
076	4C	L	108	6C	L
077	4D	M	109	6D	M
078	4E	N	110	6E	N
079	4F	O	111	6F	O
080	50	P	112	70	P
081	51	Q	113	71	Q
082	52	R	114	72	R
083	53	S	115	73	S
084	54	T	116	74	T
085	55	U	117	75	U
086	56	V	118	76	V
087	57	W	119	77	W
088	58	X	120	78	X
089	59	Y	121	79	Y
090	5A	Z	122	7A	Z
091	5B	[(left/opening bracket)	123	7B	{ (left/opening brace)
092	5C	\ (back slash)	124	7C	| (vertical bar)
093	5D	] (right/closing bracket)	125	7D	} (right/closing brace)
094	5E	^ (caret/cirumflex)	126	7E	~ (tilde)
095	5F	_ (underscore)	127	7F	(delete)