NHL STENDEN
computer vision & data science

Computer Vision

VisionLab development environment

10 April 2018

j.van.de.loosdrecht@nhl.nl, jaap@vdlmv.nl

Overview

- Development environments (general)
- VisionLab
  - Overview of system
  - Image types
  - Co-ordinate system
  - Dyadic operators
  - Displaying of images
  - Using camera's
  - Tethering with DSLR camera's
  - File Drop Cam (*)
  - File Cam (*)
  - Image Logger (*)
  - Online help
  - Widgets
  - Script language
  - Speeding up scripts (*)
  - Adding user script as operator (*)
  - Adding user C++ operators (*)
  - Adding a user plugin with new operators (*)
  - Adding a C++ operator to the VisionLab server (*)
  - Using VisionLab command interpreter in #C (*)

8/27/2018 VisionLab 2

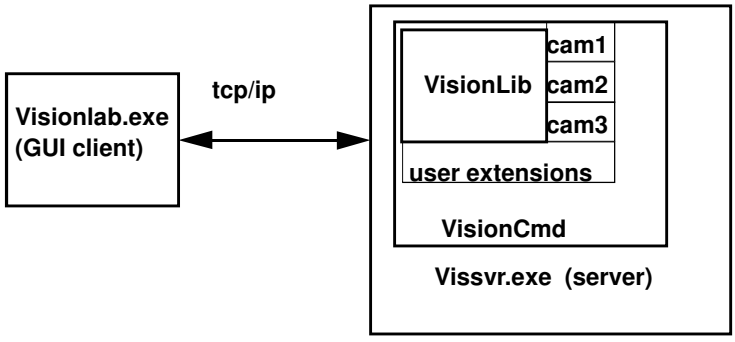
Development environments (general)

- Image acquisition
- Library with standard operators
- Experiment with operators
  - Interactive
  - Scripts
  - Graphical programming
- Add user defined operators
- Stand-alone applications

- Survey of 10 image processing packages by NHL and TNO TPD

8/27/2018 VisionLab 3

VisionLab Architecture



Demo version can be downloaded from www.vdlmv.nl (Windows and Linux)

8/27/2018 VisionLab 4

VisionLab Architecture (*)

- **VisionLib:**
  Library of more the 200 operators for image processing, classifying with neural networks and interfaces to cameras, written in ANSI C++.
- **VisionCmd:**
  Command interpreter for scripts. The scripts can be used for easy and fast development of applications or prototypes.
- **VisionServer:**
  Shell with a network interface around VisionCmd. With VisionServer it is possible to communicate over the internet with a remote VisionCmd.
- **VisionClient:**
  Graphical user interface with which it is possible to experiment in a comfortable way with the VisionLib library and to develop and test scripts. An online help is available.
- **VisionLibDLL:**
  Dll library around VisionCmd, making it accessible other languages. Delphi wrapper is available.
- **VisionLibNet:**
  .NET class wrapper around VisionLibDLL, making it accessible by all .NET languages.
- **VisionLibPugIn:**
  PlugIn system for dlls to extend the command interpreter.

8/27/2018 VisionLab 5

Demonstration VisionLab

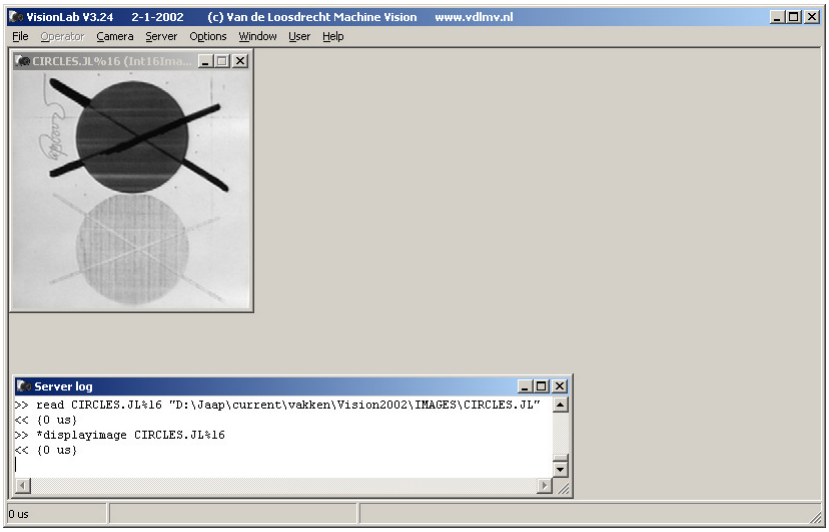
- **Start-up VisionLab**
- **Show console server**
- **Explain Server Log window of client**
- **Open circles.jl and show server/client communication**
- **Demonstrate opening of *.bmp image, is automatically converted to RGB888Image, use dark_flower.bmp in images directory**

- ***NOTE:***
  VisionLab does NOT allow for space in names of directories and file names!!

- **Source image circles.jl: Lex van de Voort van de Kleij (NHL)**

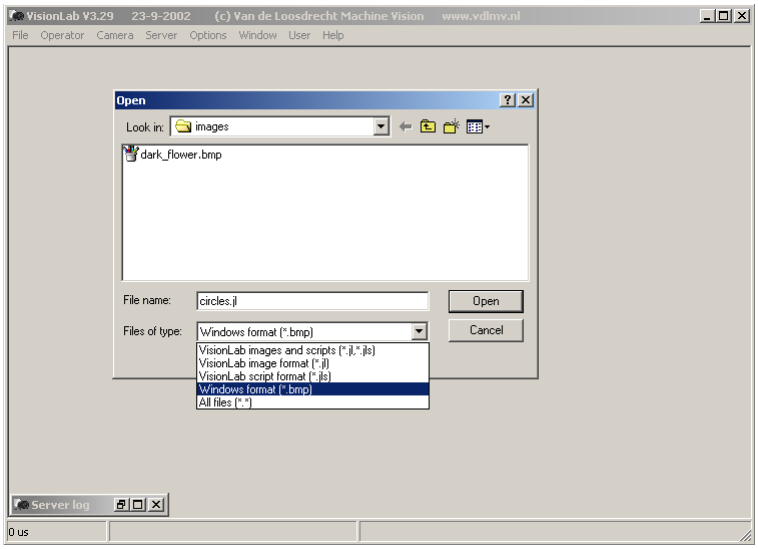
8/27/2018 VisionLab 6

Demonstration VisionLab



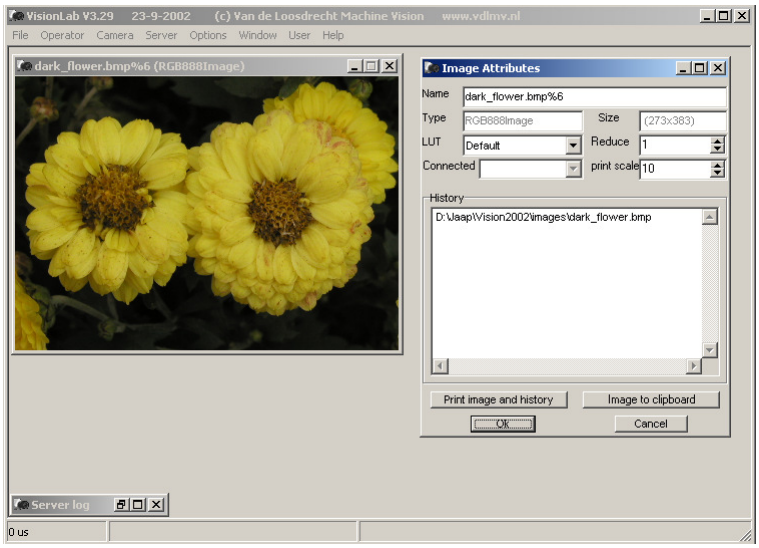
8/27/2018 VisionLab 7

Open *.bmp image



8/27/2018 VisionLab 8

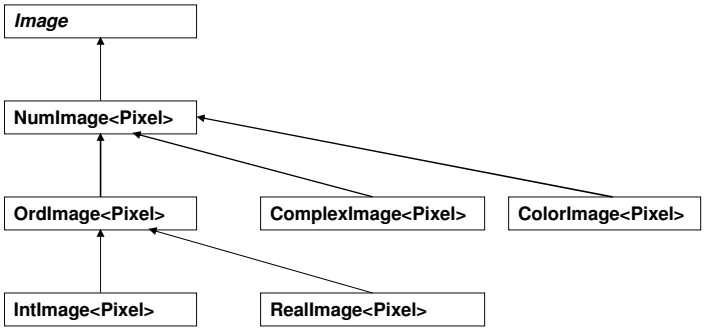
Conversion to RGB888Image



8/27/2018 VisionLab 9

Image types (*)

Class hierarchy using templates:



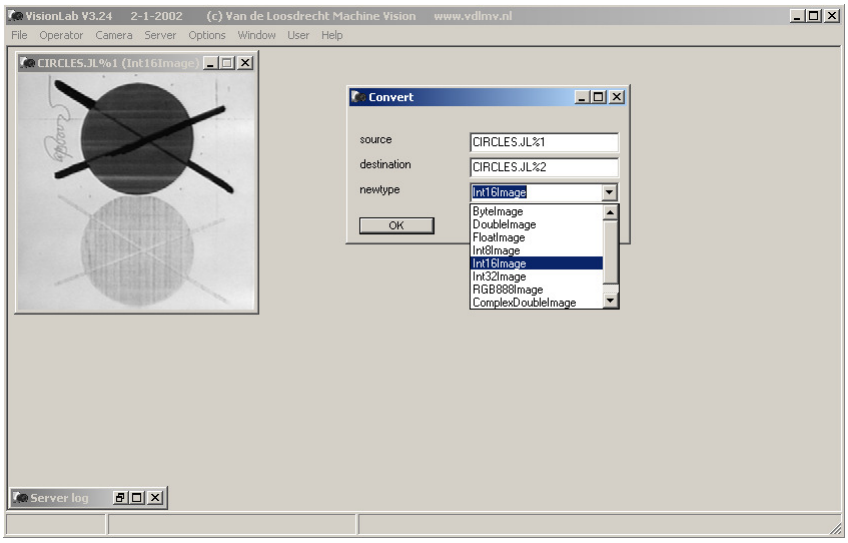
8/27/2018 VisionLab 10

Image types

- ***Image***
  - **NumImage<Pixel>:**
    - **OrdImage<Pixel>:**
      - **IntImage<Pixel>:**
        - **ByteImage**
        - **Int8Image**
        - **Int16Image**
        - **Int32Image**
      - **RealImage<Pixel>:**
        - **FloatImage**
        - **DoubleImage**
    - **ColorImage<Pixel>:**
      - **RGB888Image, RGB161616Image**
      - **HSV888Image, HSV161616Image**
      - **YUV888Image, YUV161616Image**
    - **ComplexImage<Pixel>:**
      - **ComplexFloatImage**
      - **ComplexDoubleImage**

8/27/2018 VisionLab 11

Conversion between Image Types



8/27/2018 VisionLab 12

Co-ordinate system and Dyadic operators

Co-ordinate system

- **Origin: top left corner**
- **X values increase from left to right**
- **Y values increase from top to bottom**

Dyadic operators:

- **Use select second**

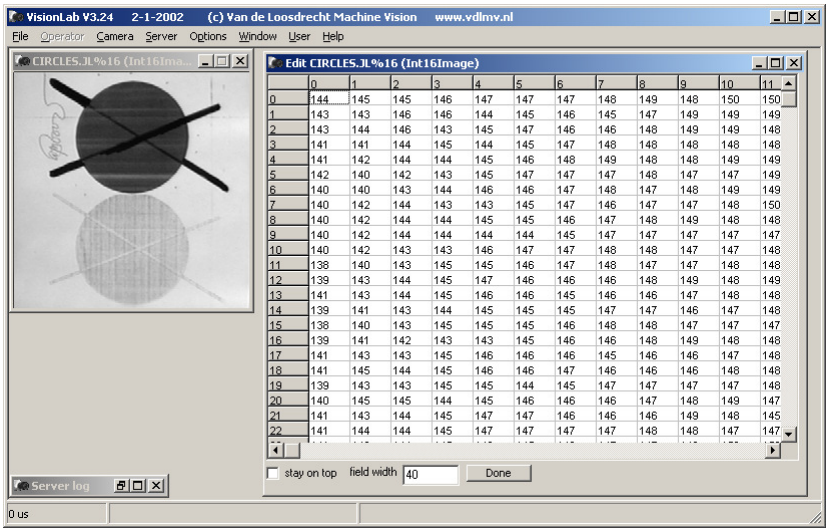
8/27/2018 VisionLab 13

Demonstration co-ordinate system and DisplayLUTs

- **Show co-ordinate system with Edit**
- **Demonstrate select second :**
  - **DisplayLUT for Int16Image = Stretch**
  - **Copy circles**
  - **Add the two images**
  - **No visible difference!**
- **Show pixel values of both images with Edit**
- **Explanation in slide about DisplayLUTs**

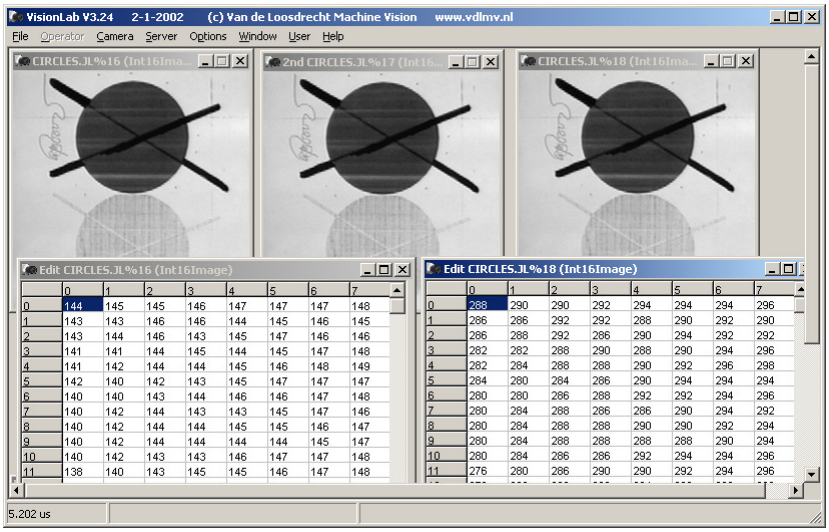
8/27/2018 VisionLab 14

Demonstration co-ordinate system



8/27/2018 VisionLab 15

Add images using select 2nd



8/27/2018 VisionLab 16

Displaying of images

Server:

- **List of images on which operations act**

GUI client:

- **Display image with use of LookUp Table (LUT)**

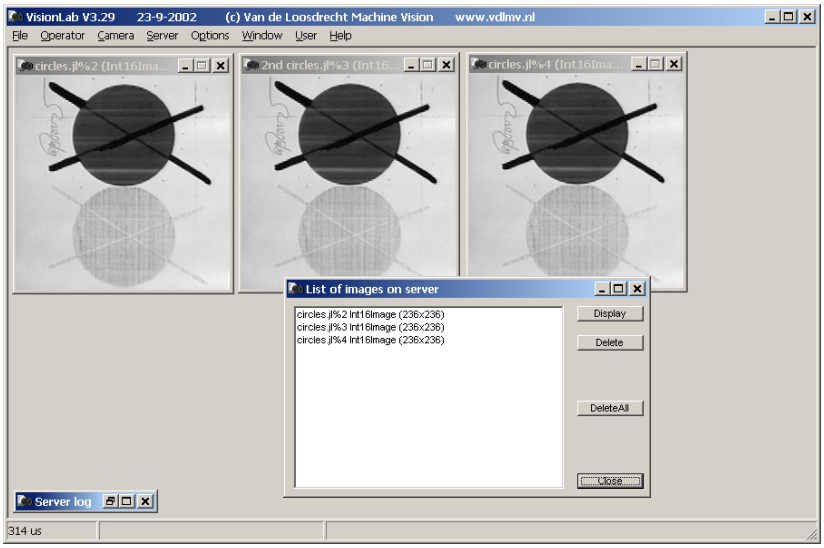
8/27/2018 VisionLab 17

Demonstration of displaying of images

- **Show list of images on server**
- **Client is 'stupid', it knows nothing about vision operators**
- **Client can only translate commands and display images**
- **All vision intelligence is in the server**
- **Demonstrate applying different LUT's on image, see next slide**

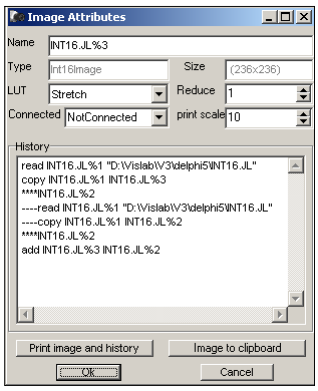
8/27/2018 VisionLab 18

List of images on server



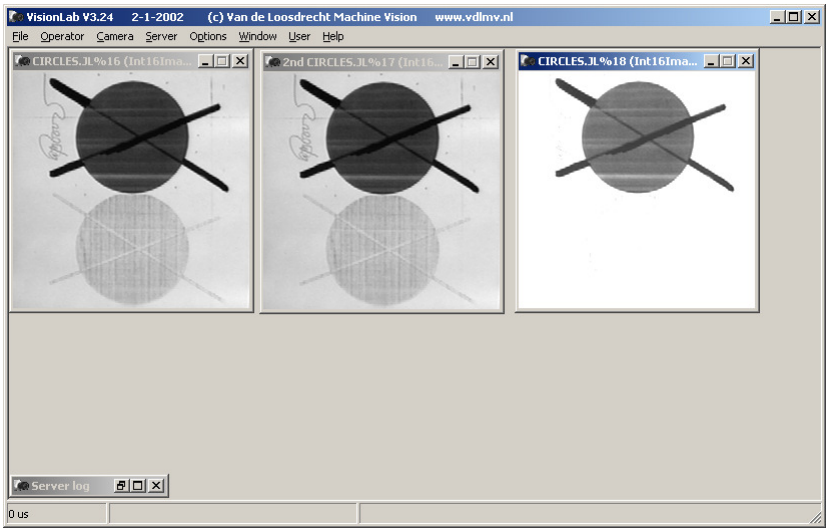
8/27/2018 VisionLab 19

Set LUT for result image to Clip



8/27/2018 VisionLab 20

See result of adding the two images



8/27/2018 VisionLab 21

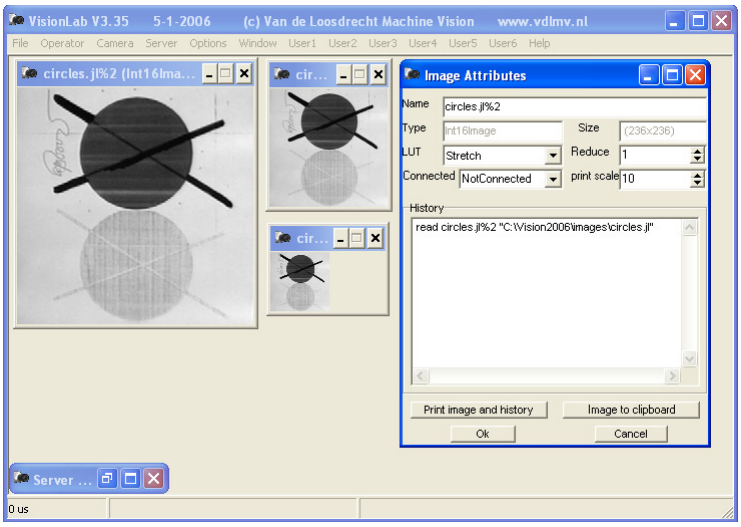
Displaying of images

DisplayLUT's (LUT = LookUpTable):

- **IntImage**
  - **Stretch**
  - **Clip**
  - **Labelled**
  - **Binary**
- **FloatImage**
  - **Linear**
  - **Logarithmic**
- **ColorImage**
  in colour depending on screen settings
- **ComplexImage**
  power spectrum displayed as FloatImage

8/27/2018 VisionLab 22

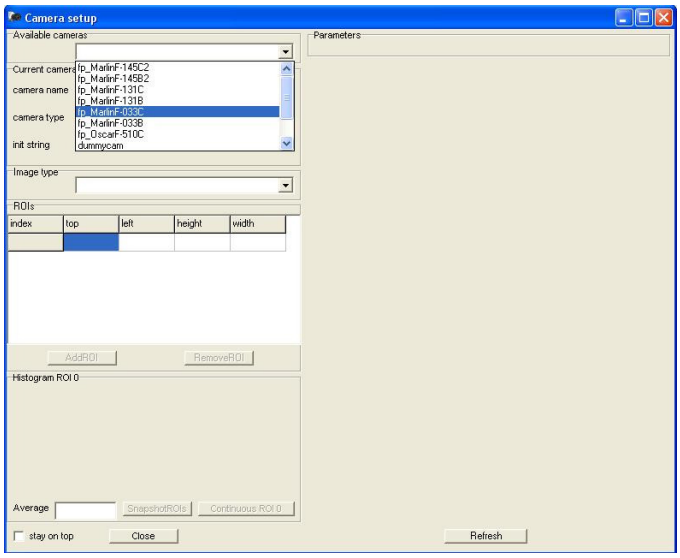
Change reduce factor with scroll wheel or + or - button



Camera setup

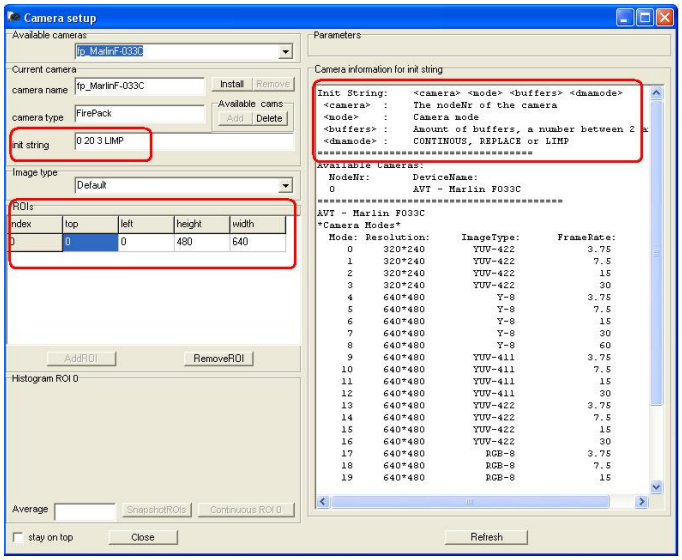
- **Multiple camera's**
- **Multiple windows (ROI's) per camera**
- **Dynamic loading of parameters**
- **Support for:**
  - **Fire wire (FirePackage)**
  - **IDS uEye**
  - **XIMEA**
  - **File cam**
  - **Microsoft Media Foundation Interface web cams**
- **Open camera interface,**
  easy to add new camera's or frame grabbers
- **Camera recorder**

Select camera



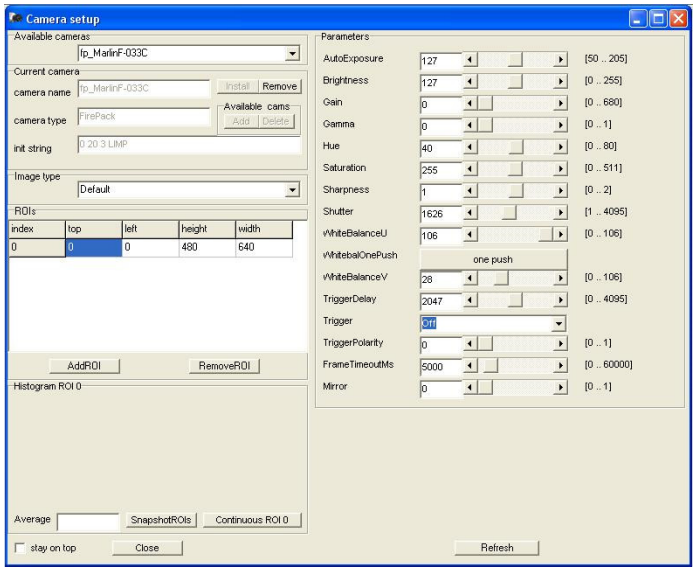
8/27/2018 VisionLab 25

Set init string, select mode



8/27/2018 VisionLab 26

Ready for taking snapshot



8/27/2018 VisionLab 27

Exercise

- **Experiment with camera setup**

8/27/2018 VisionLab 28

Tethering with DSLR camera's (*)

Using the middleware from digicamcontrol.com (open source) most Nikon and Canon DSLR's can be tethered from VisionLab

Example:

- **Install digicamcontrol**
- **Use script below to acquire image from DSLR:**

```
// Note: set camera in jpeg mode!
$path = lpwd
$slash = "\"
$fileName = snapshot.jpg
$fileName = $path . $slash . $fileName
system "C:\Program Files (x86)\digiCamControl\CameraControlCmd.exe" /filename $fileName /capture
lread i $fileName
display i
```

8/27/2018 VisionLab 29

File Drop Camera (*)

Used as a rudimentary interface to cameras who are able to write bmp or jpg images to file with a file name and a sequence number.
The file name consists of a <File base name> + <sequence nr> + . + <File extension>. The sequence nr is padded with leading 0's if <nr of digits> is greater then 0.

After pressing the start button the image with the specified filename is searched for. If the specified file is not found, processing is postponed until the specified file is created.

If the file is found or created:
- the image is opened and displayed
- the image is selected as script image (%current image)
- the script with <Script name> is executed with the specified <Parameters> the function result of the script is assigned to the specified <Result> variable
- <Sequence nr> is incremented
- the search or wait for the next image starts

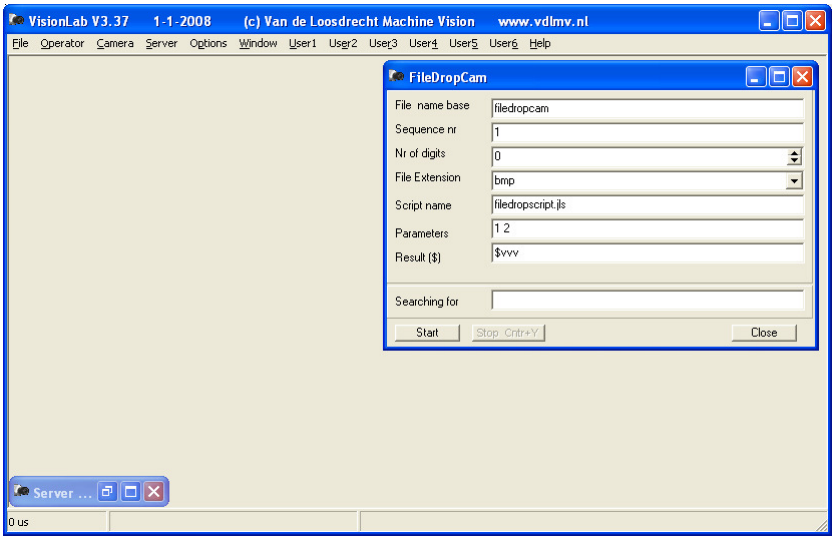
8/27/2018 VisionLab 30

Demonstration File Drop Camera (*)

- Move file filedropcam2.bmp from directory to the desktop, leave filedropcam1.bmp in current directory
- Open File Drop Cam form in Camera menu
- Specify the parameters as in the next slide
- Press start button
- File filedropcam1.bmp is in current directory, so the image is opened and selected as %currentimage and script filedropscript.jls is executed
- Because file filedropcam2.bmp is in not in the specified directory, processing is postponed.
- Drop file filedropcam2.bmp back in specified directory, processing continues, image is opened and script is executed

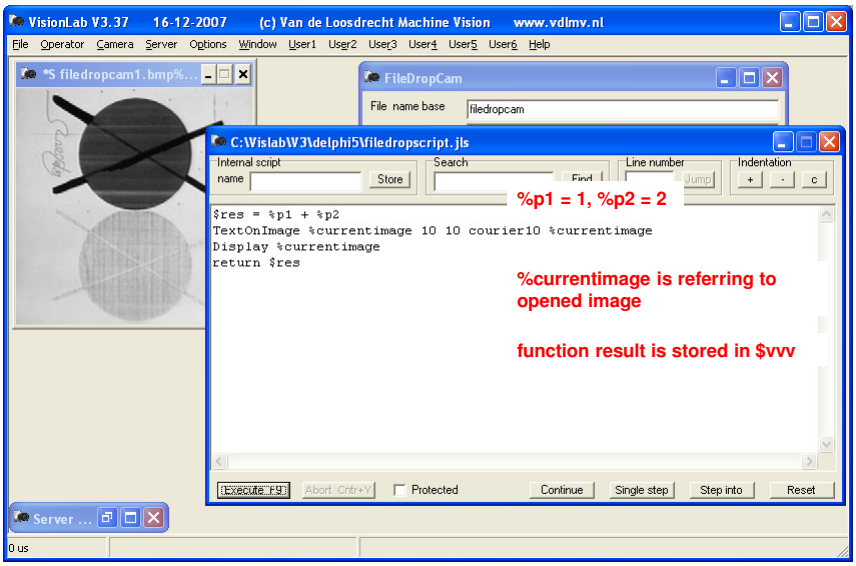
8/27/2018 VisionLab 31

File Drop Cam form in Camera menu (*)



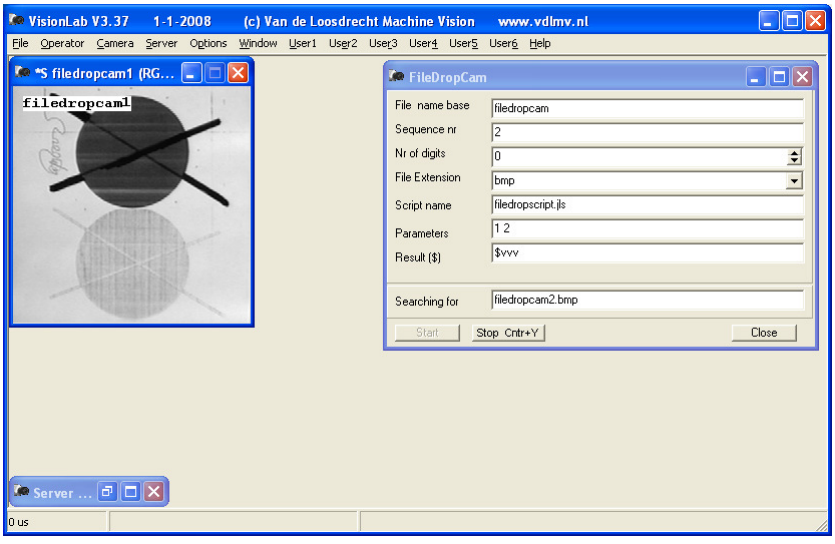
8/27/2018 VisionLab 32

Image is opened %currentimage and script is executed (*)



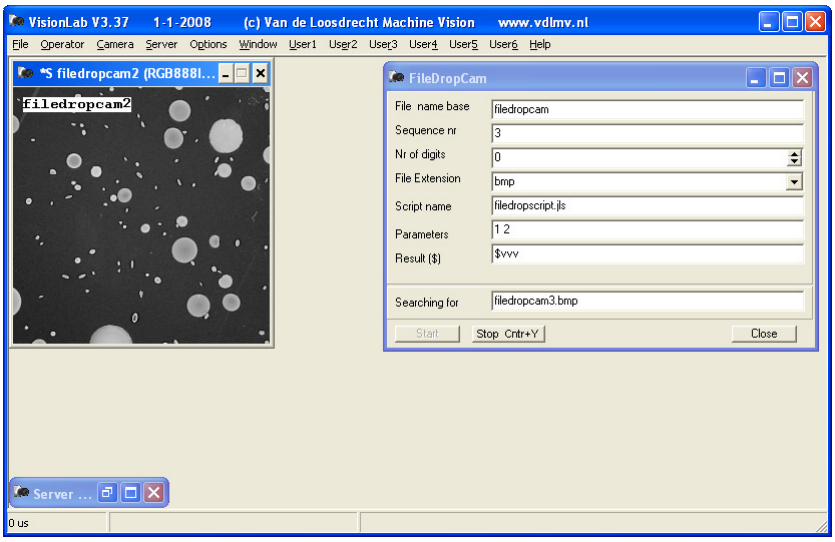
8/27/2018 VisionLab 33

Wait for drop file filedropcam2.bmp back in specified directory (*)



8/27/2018 VisionLab 34

File dropcam2.bmp is processed (*)



8/27/2018 VisionLab 35

FileCam (*)

FileCam is used to 'snapshot' images from a file from disk.

The init string should be used to specify in which mode the camera is operated. Syntax:

<imageType > <fileName>

The file is in normal text file with on each line the file name of an jl image. It is adviced to use a fully specified file names including the full direcory path.

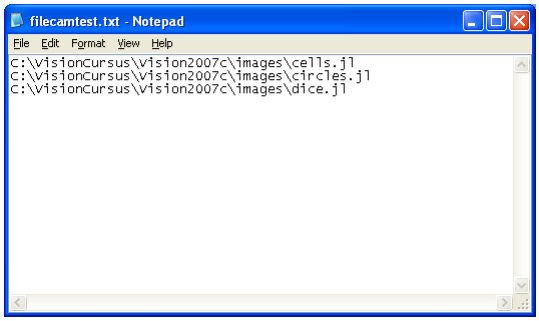
Snapshoting will start from the first image in the file.

If more snapshots are taken then images specified in the file, snapshoting will start again from the first image

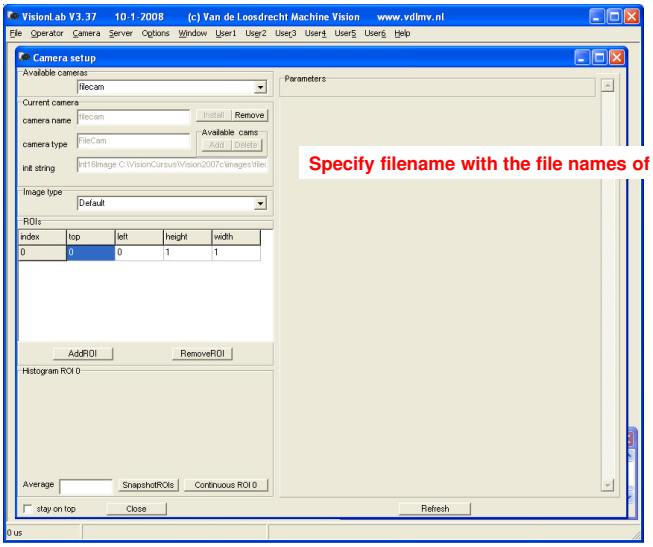
Use the command LastFileCamName to retrieve the fileName of the last snapshot image.

8/27/2018 VisionLab 36

Example: using a FileCam (*)



Example: using a FileCam (*)



Example: using a FileCam (*)

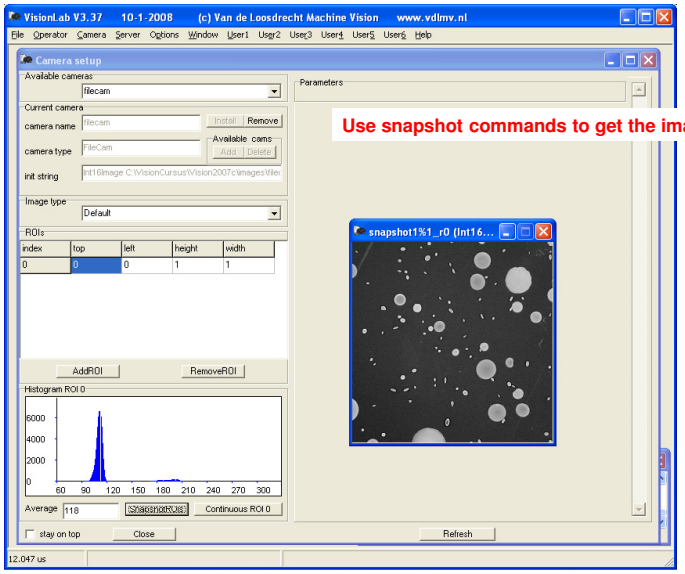


Image Logger (*)

Intended to log images with additional string information

During an experiment images with string information are stored in RAM memory. This will give only a slow delay in execution.
After the experiment is finished, the images and strings can be stored to files for subsequent analysis.

Operators:

- **InitImageLogger**
- **CloseImageLogger**
- **ImageLoggerAdd**
- **ImageLoggerClear**
- **ImageLoggerSaveToFile**
- **ImageLoggerGetNth**

Image Logger (*)

InitImageLogger imageType loggerType maxNrFrames height width maxStrSize disgard

The InitImageLogger operator initialise the image logger. RAM memory for maxNrFrames of image of imageType and maxNrFrames of strings of size maxStrSize is reserved.

The loggerType parameter can be LinearImageLogger or CircularImageLogger. When the log of LinearImageLogger is full new added images and strings are ignored. When the log of a CircularImageLogger is full the oldest image and string are deleted and the new image an string are added.

The disgard parameter specifies how many ImageLoggerAdd commands are ignored before an image and string are stored in RAM memory.

CloseImageLogger

The CloseImageLogger operator releases all allocated memory.

8/27/2018 VisionLab 41

Image Logger (*)

ImageLoggerAdd imageName loggerMode string

The ImageLoggerAdd operator adds an image and string to the log.

The parameter loggerMode can have the values DisgardNImageLogger or ForceImageLogger.

In DisgardNImageLogger mode disgard images are ignored before an image is stored in the log.

In ForceImageLogger mode the image and string is always stored.

ImageLoggerClear

The ImageLoggerClear operator clears all logged images and strings.

8/27/2018 VisionLab 42

Image Logger (*)

ImageLoggerSaveToFile fileNameBase

The ImageLoggerSaveToFile operator save all image to file.

The parameter fileNameBase is used to generate unique file names. The Nth image is saved as fileNameBase_N.jl.

The strings are saved in a file with name fileNameBase.txt. The format of fileNameBase.txt is one line for each image: <fileName> <string>.

ImageLoggerGetNth nth imageName

The ImageLoggerGetNth operator retrieves the nth image in the log and makes a copy into image with imageName.

The function result is the string associated string with the Nth image.

8/27/2018 VisionLab 43

Image Logger example script (*)

```
$path = lpwd
$oldpath = pwd
cwd $path
system del test__log*.*
InitImageLogger Int16Image CircularImageLogger 10 512 512 100 2
Create image Int16Image 512 512
SetAllPixels image 100
for $i = 1 to 25 do
   TextOnImage image 0 40 courier14 $i
   ImageLoggerAdd image DisgardNImageLogger $i
endfor
ImageLoggerSaveToFile test__log
$r = ImageLoggerGetNth 1 test
Display test
CloseImageLogger
cwd $oldpath
= $r
```

8/27/2018 VisionLab 44

Online help

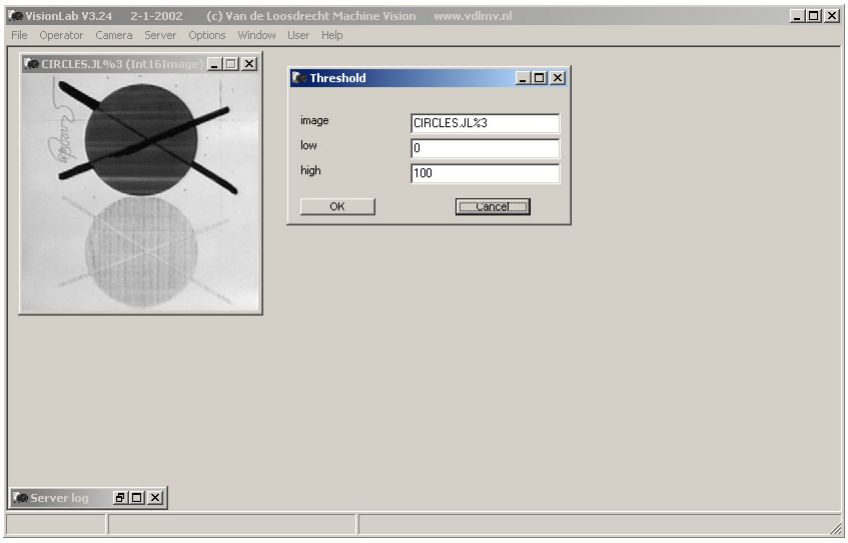
Online help is available:

- **F1**
- **Main menu**

Demonstration online help

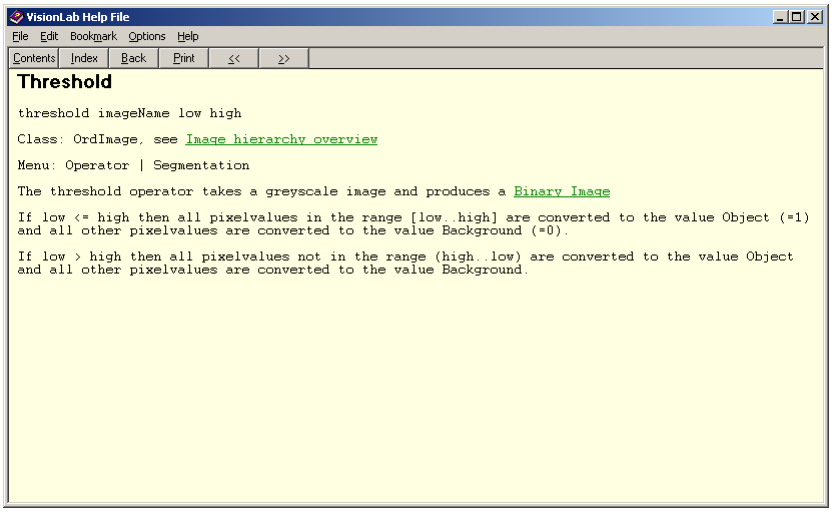
- **Search for Threshold operator**
  - **Open image circles.jl**
  - **Select threshold operator and press F1 for online help**
  - **description operator and position in operator menu structure**
  - **Select help from main menu**
  - **Select Find in VisionLab help file window and type threshold**
  - **C++ header file in System Development Kit**

Open image and select threshold operator



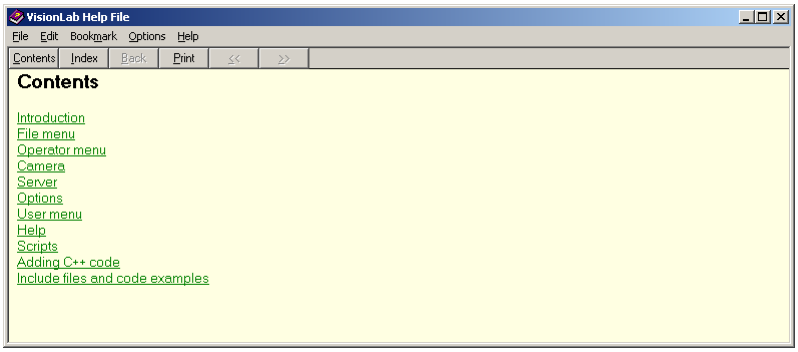
8/27/2018 VisionLab 47

Press F1 for help for Threshold (GUI)



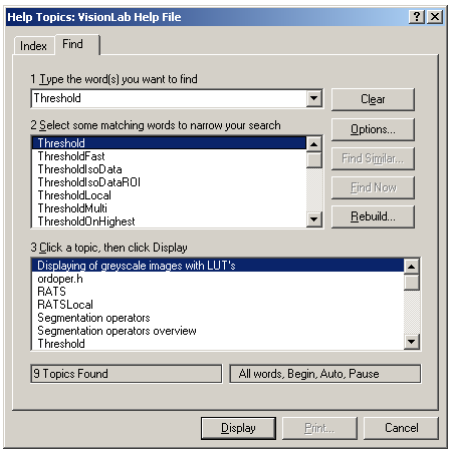
8/27/2018 VisionLab 48

Select help from main menu



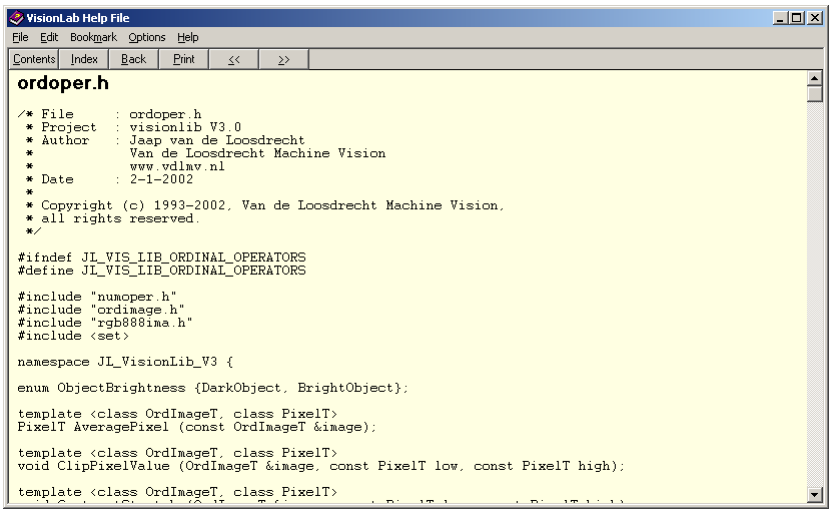
8/27/2018 VisionLab 49

Select Find in VisionLab help file window

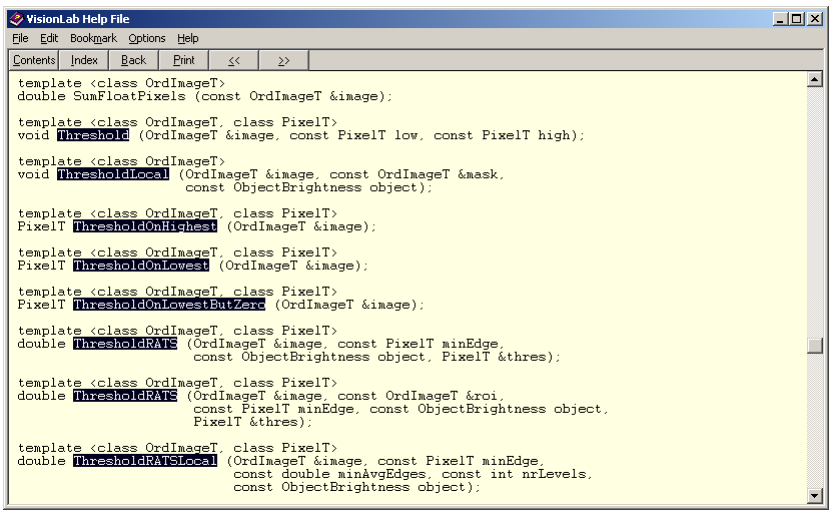


8/27/2018 VisionLab 50

Help for Threshold (SDK) (*)



Help for Threshold (SDK) (*)



Widgets

Several operators need as parameter one or more pixel coordinates. The widget tools will help the user to specify the pixel coordinates in an interactive way.

Before selecting the desired operator, the widget tool must be selected. This tool will display a grid on the image. The grid can be manipulated by dragging and rotating the landmarks. After the grid is positioned at the desired position the operator is called from the menu. The landmarks are then extracted from the grid and displayed the corresponding parameter fields.

Menu: Operator | Widget tools

8/27/2018 VisionLab 53

Widgets

Available widgets tools:

- **LineTool: start coordinate and end coordinate of a line**
- **GridTool: four corner coordinates of deformable rectangle**
- **RasterTool: middlePoint coordinate, endLine coordinate, endBox coordinate for scanlines**
- **CircleTool: center coordinate of circle and radius of circle**
- **PieTool: center coordinate of pie, start angle, end angle and radius of pie**
- **MinMaxCircleTool: center coordinate of circles, minRadius and maxRadius of circles**

8/27/2018 VisionLab 54

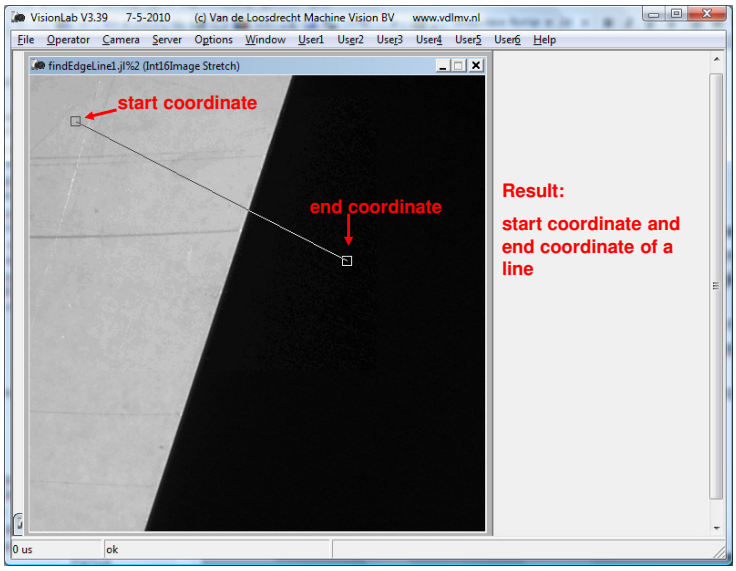
Widgets

Operators using widgets:

CircleShape	**CircleTool**
DiskShape	**CircleTool**
DrawLine	**LineTool**
FindEdgeCircle	**MinMaxCircleTool**
FindEdgeLine	**RasterTool**
FindSubEdge(s)OnLine	**LineTool**
HoughCircleTransforms	**MinMaxCircleTool**
HoughLineTransforms	**PieTool**
PolarStretch	**PieTool**
ResampleLine	**LineTool**
Warp	**GridTool**

8/27/2018 VisionLab 55

Line tool

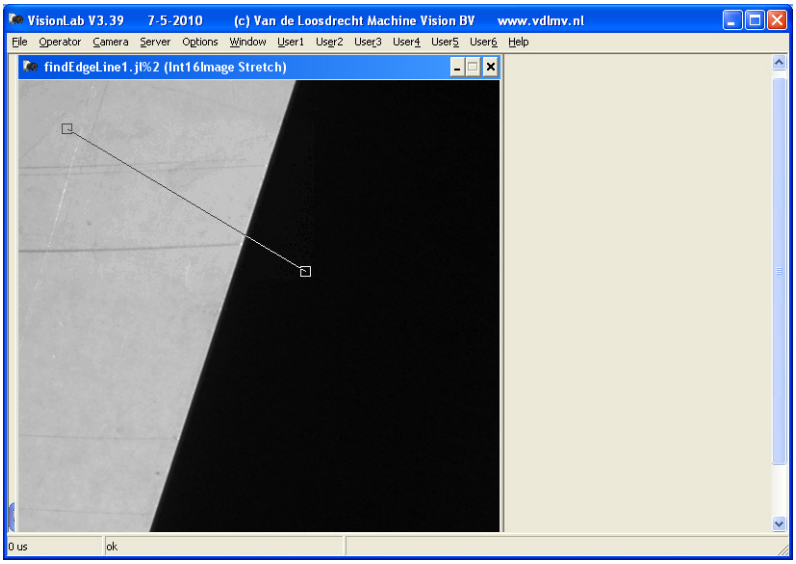


8/27/2018 VisionLab 56

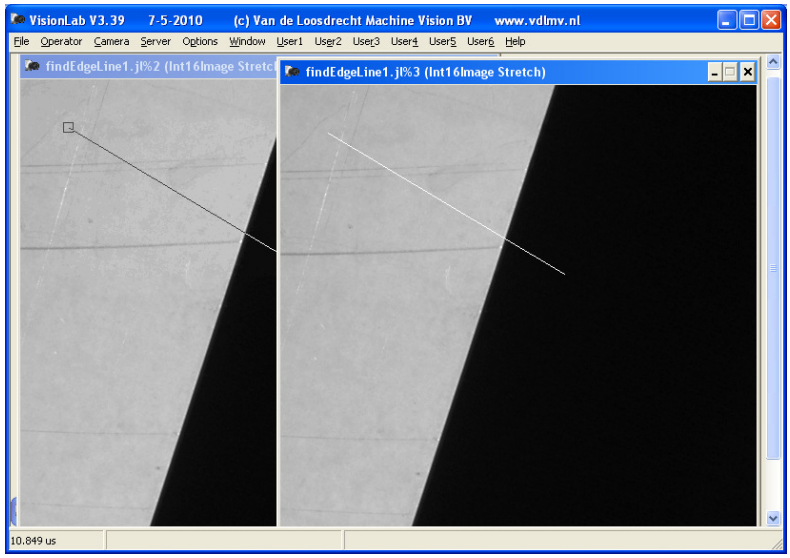
Demo line tool

- **Open image findEdgeLine1.jl**
- **Open grid tool from menu Operator | Widget tools | DrawLine**
- **Drag landmarks to position the line**
- **DrawLine operator from menu Operator | Syntetic**

Drag landmarks to position the line

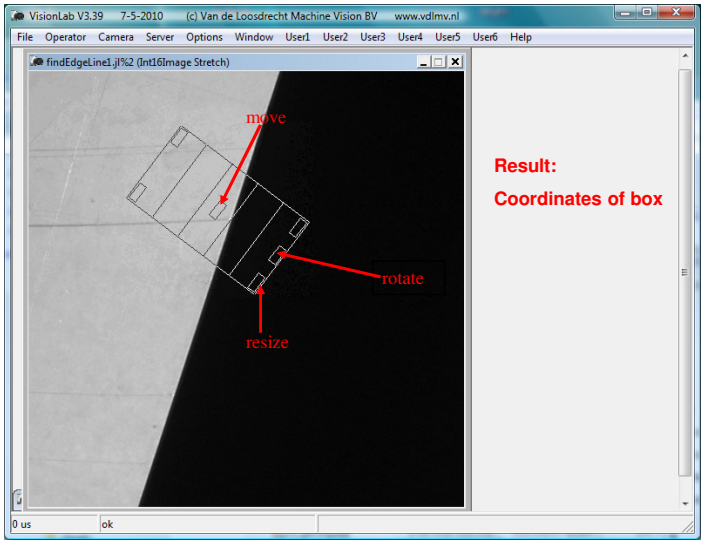


DrawLine operator



8/27/2018 VisionLab 59

Raster tool



8/27/2018 VisionLab 60

Demo raster grid tool

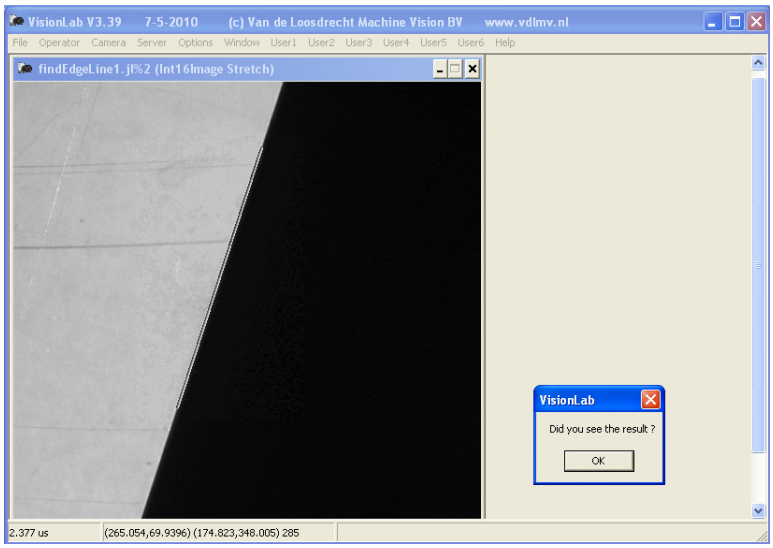
- **Open image findEdgeLine1.jl**
- **Open grid tool from menu Operator | Widget tools | FindEdgeLine**
- **Drag landmarks to position**
- **Select FindEdgeLine operator from menu Operator | EdgeDetection (line distance = 1, outlierDistance = 10, iteration = 10)**

8/27/2018 VisionLab 61

Drag landmarks to position

8/27/2018 VisionLab 62

FindEdgeLine operator



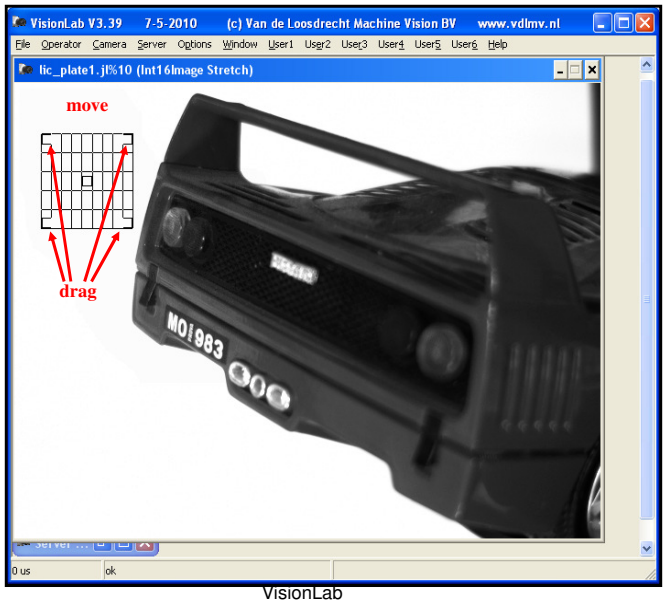
8/27/2018 VisionLab 63

Demo grid tool

- **Open image lic_plate1.jl**
- **Open grid tool from menu Operator | Widget tools | Warp**
- **Drag landmarks to corners of license plate**
- **Select Warp operator from menu Operator | Geometry**

8/27/2018 VisionLab 64

Grid tool



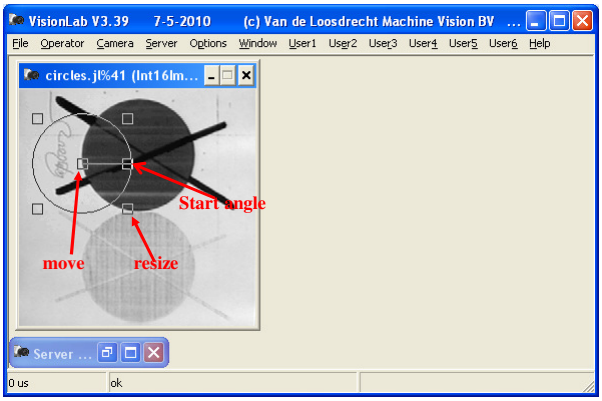
8/27/2018 VisionLab 65

Grid tool: drag landmarks to corners

8/27/2018 VisionLab 66

Grid tool: correct for warp

Circle tool

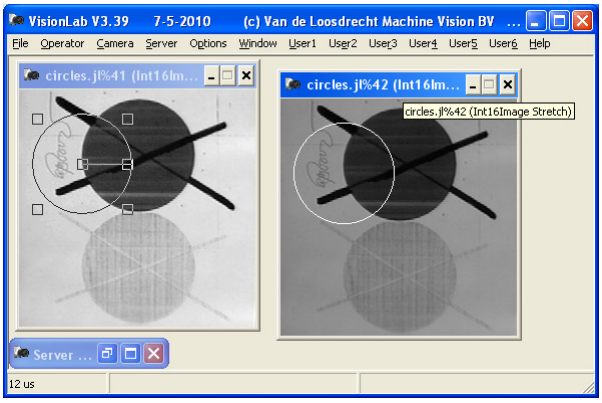


Demo circle tool

- **Open image circles.jl**
- **Open circle tool from menu Operator | Widget tools | CircleAndDiskShape**
- **Drag landmarks to desired position**
- **Select CircleShape operator from menu Operator | Synthetic (value = 255, zeroOrOrg = KeepOriginal)**

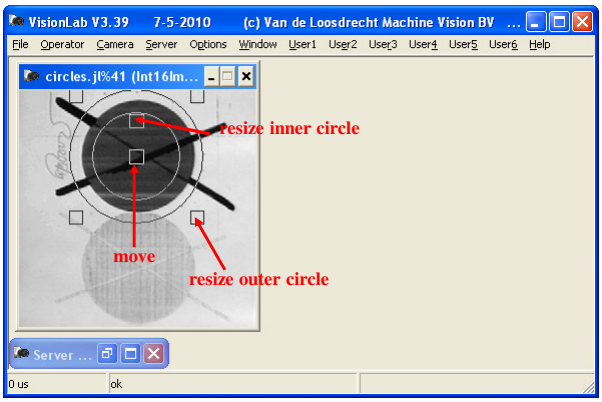
8/27/2018 VisionLab 69

Circle tool: CircleShape



8/27/2018 VisionLab 70

MinMaxCircleTool: two concentric circles



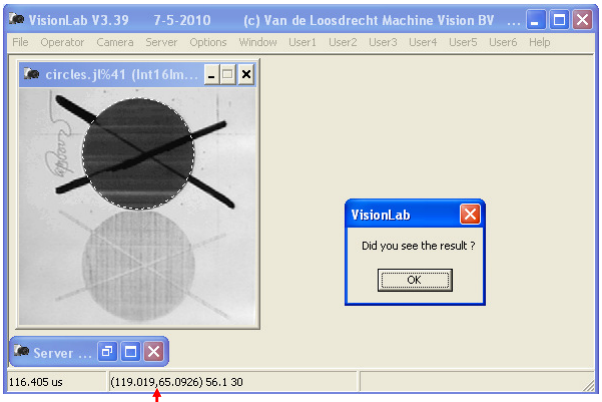
8/27/2018 VisionLab 71

Demo min max circle tool

- **Open min max circles tool from menu Operator | Widget tools | HoughCircles**
- **Drag landmarks to desired position**
- **Select FindFastBestCircle operator from menu Operator | Transforms (minEdge = 500, deltaR = 0.1)**

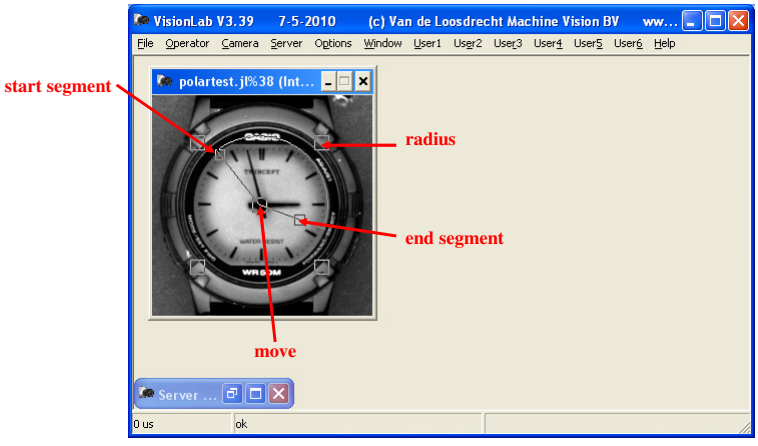
8/27/2018 VisionLab 72

MinMaxCircleTool: FindBestCircle with minR .. maxR



8/27/2018 VisionLab 73

Pie tool: slice of a circle



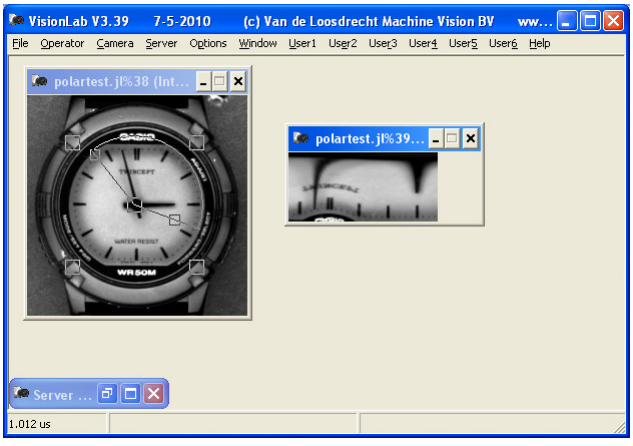
8/27/2018 VisionLab 74

Demo pie tool

- **Open pie tool from menu Operator | Widget tools | PolarStretch**
- **Drag landmarks to desired position**
- **Select PolarStretch operator from menu Operator | Transforms (minEdge = 500, deltaR = 0.1)**

8/27/2018 VisionLab 75

Operator PolarStretch



8/27/2018 VisionLab 76

Script language

Scripts can contain different kind of commands:

- **image operators**
- **variables and expressions (*)**
- **control statements (*)**
- **internal commands (*)**
- **Speeding up scripts (*)**
- **special server commands (*)**
- **pre-processor commands (*)**

Calling scripts (procedure/function call): (*)

Scripts can be added as new operators to GUI (*)

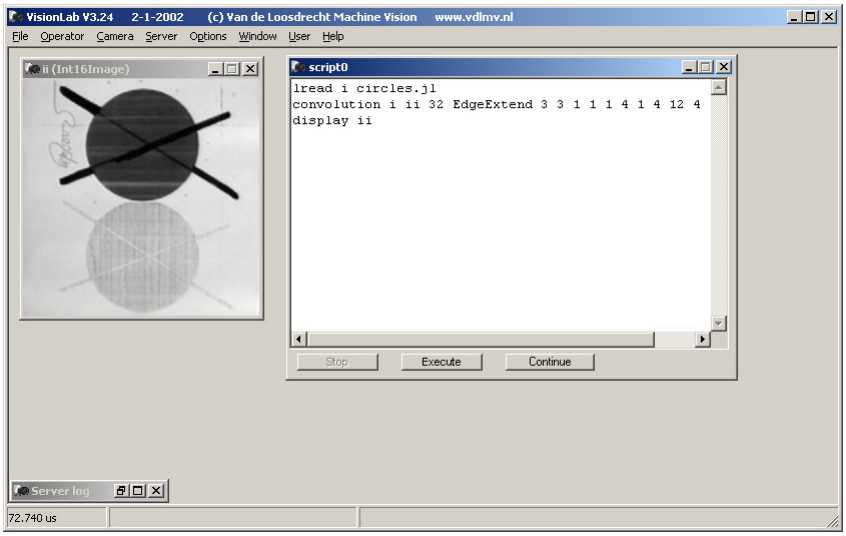
8/27/2018 VisionLab 77

Demonstration script image operators

- **Create new script, add the lines by selecting the operators from the menus (not necessary to type them):**
  - **Iread i circles.jl**
  - **display i**
- **Execute script**
- **Insert with mouse threshold command, and adapt 'imageName', explain why**
- **Execute script**
- **Demonstrate convolution instead of threshold:**
  - **convolution i ii gausian3x3**
  - **display ii**
- **Change to SobelNS by doubleclick on convolution line in script**

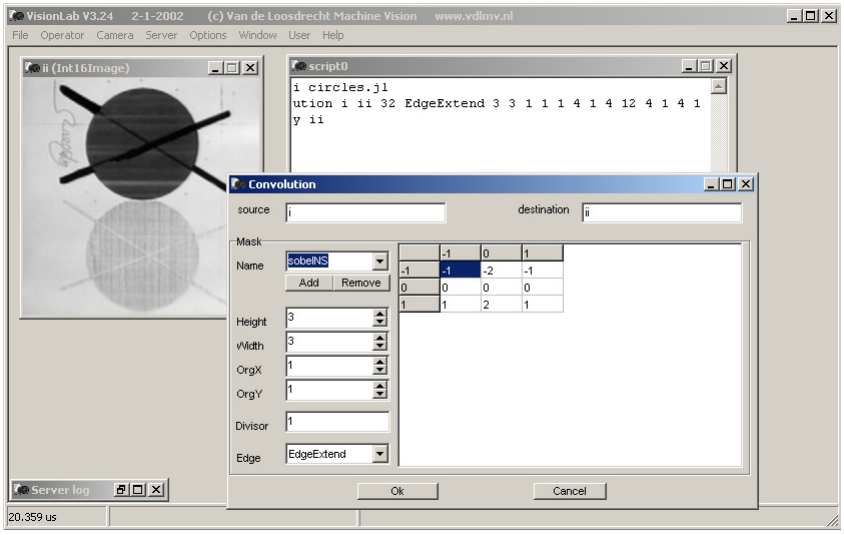
8/27/2018 VisionLab 78

Example script



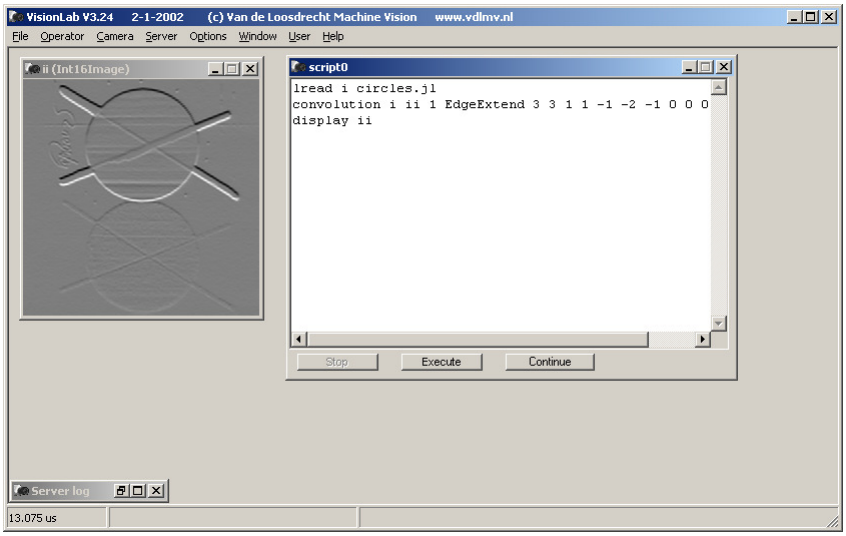
8/27/2018 VisionLab 79

Double click on line with convolution and …



8/27/2018 VisionLab 80

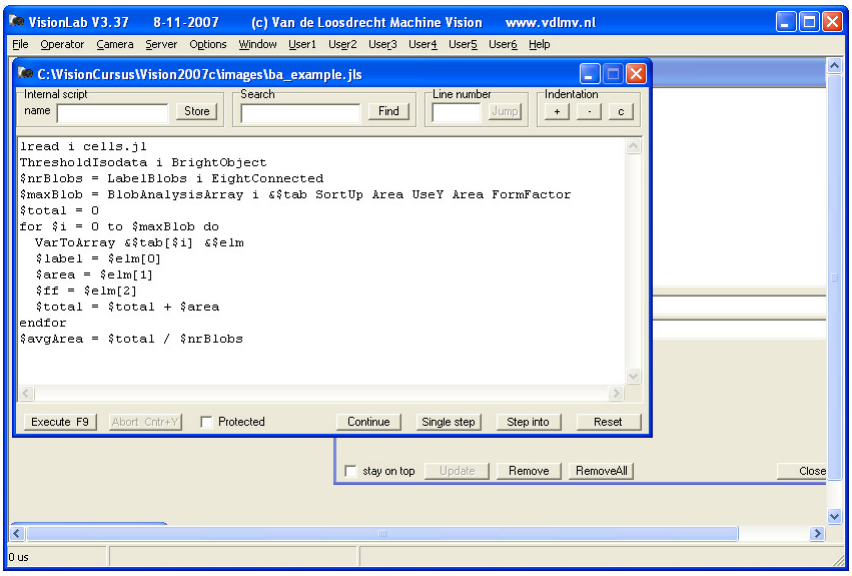
… select mask Sobel NS



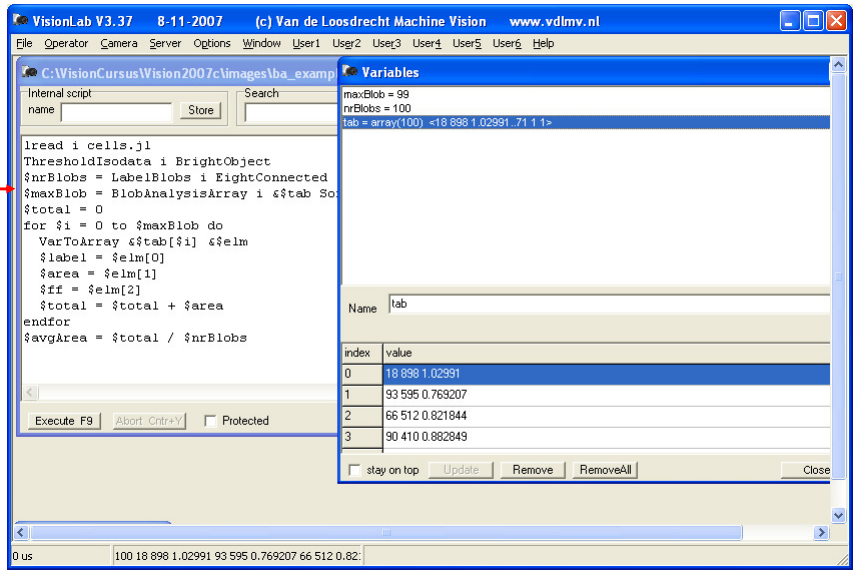
Demonstration script variables and control statements

- This script calculates the average area of the selected blobs
- Open scripts ba_example.jls
- Open variable screen (Server menu | Examine variables)
- Single step through script
- Notes:
  - result is returned to an array with name tab
  - Click on array name in top window of variable screen to examine details of array
  - each element of the array contains a line with:
    <labelnr> followed with the specified measurements
  - Each line is extracted from the array tab to an array elm
  - The element with index 0 of array elm is the labelnr
  - The element with index 1 of array elm is the area
  - The element with index 2 of array elm is the formfactor

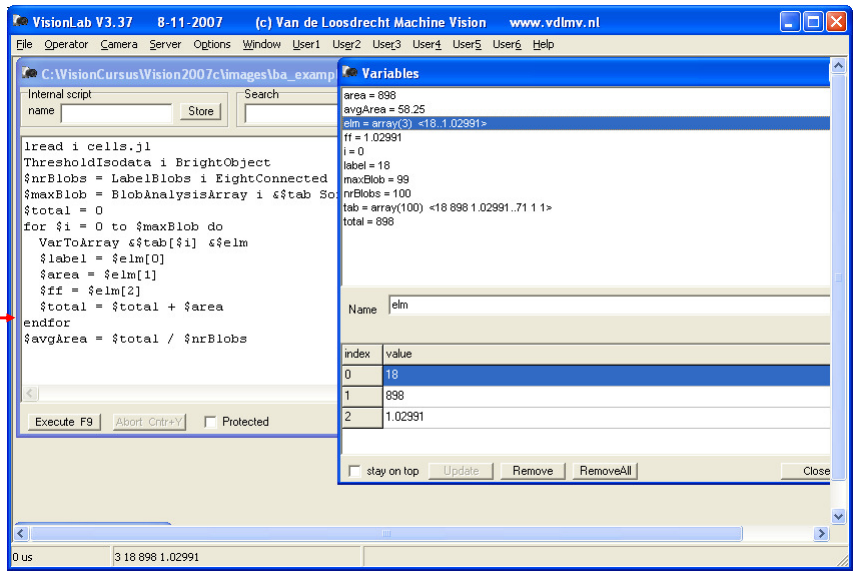
Demonstration script variables and control statements



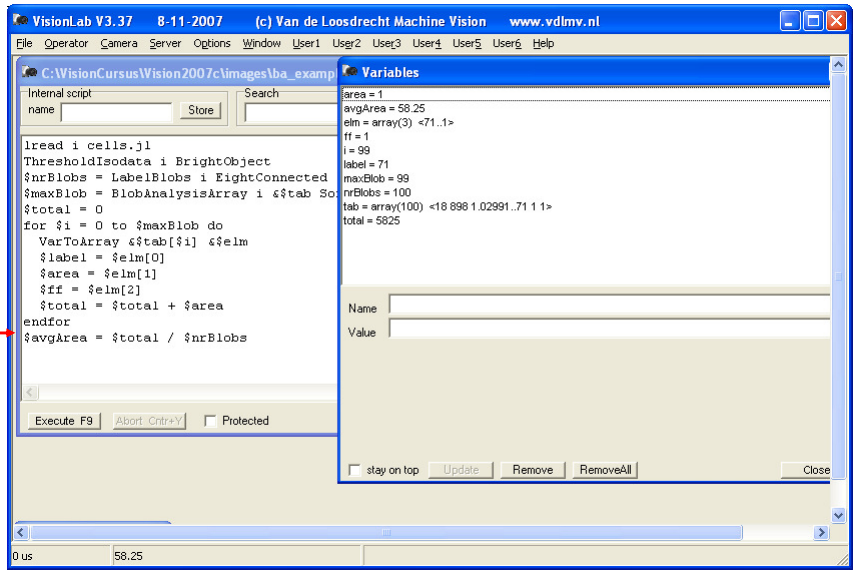
Demonstration script variables and control statements



Demonstration script variables and control statements



Demonstration script variables and control statements



Exercise

- **Experiment with scripts**

Script variables (*)

Variables are of type string. When necessary a conversion is applied to type integer, float or boolean.

There are two type of variables:

- **global: Scope is all scripts and outside the scripts. Lifetime is from creation to the moment it is removed explicitly by the user. The name of a global variable starts with one $ sign, example: $global**
- **local: scope and lifetime is the script in which the variable is created. The name of a local variable starts with two $ signs, example: $$local**

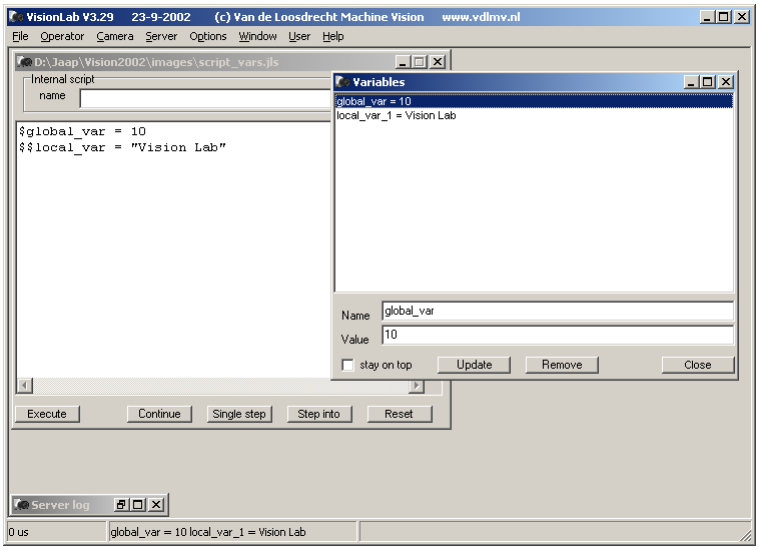
The value of the variables can be examined and modified with the 'Examine vars' item in the server menu.

Demonstration script variables (*)

- **open script script_vars.jls**
- **execute script**
- **open window 'Examine vars' in the server menu**

- **explain: Main menu | Options | Synchronize scripts**

Demonstration script variables (*)



Array's of variables (*)

Array's of variable are also possible using [and] for indexing

First index = 0
Indexes must be either constants, variables or indexed arrays
Expressions as index are not allowed
$array is a short notation for $array[0]

Example: $array[$i] = $array[$tab[0]]

8/27/2018 VisionLab 91

Array's of variables (*)

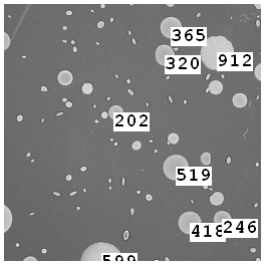
Operators for manipulation of array's:

- **ArrayToVar &$array &$var, convert array to var**
- **CopyVar &$src &$dest, copy array from src to dest**
- **ForEach &$array scriptCommand, for each element in array scriptCommand is executed. Before executing scriptCommand, occurrences of '#n' in scriptCommand are substituted by nth word in element of array**
- **GetDimVar &$array, result is dimension of array**
- **SetDimVar &$array dimension**
- **VarToArray &$var &$array, convert var to array, each word in var is new element in array**
- **ArrayToFile <fileName> &$array**
- **ArrayFromFile <fileName> &$array**
- **LArrayToFile <fileName> &$array**
- **LArrayFromFile <fileName> &$array**

8/27/2018 VisionLab 92

Example ForEach

The following script will display the size of the blobs, greater than 200 pixels, using TextOnImageCoord2D:




```
// script foreach.jls
lRead org cells.jl
Copy org label
ThresholdIsodata label BrightObject
RemoveBlobs label EightConnected Area 0 200 UseX
LabelBlobs label EightConnected
BlobAnalysisArray label &$tab SortUp Area UseY Area CentreOfGravity
ForEach &$tab TextOnImageCoord2D org #3 courier12 #2
display org
```

8/27/2018 VisionLab 93

Script expressions (*)

Simple forms of expressions are possible:

- **<var> = <var>**
- **<var> = <constant>**
- **<var> = <var> <operator <var>**
- **<var> = <var> <operator> <constant>**

Note there is not yet support for brackets ( ) in expressions.

The following operators are supported: + - * / div mod and or ! == != < <= > >= .

8/27/2018 VisionLab 94

Script expressions (*)

**Arithmetic and Boolean expressions are possible,
use brackets ( ) for priorities in Boolean expressions.**

**Note: all operators and must separated by spaces.
Examples:
$r = ($x + $v) * 4
$b = ! ($b1 and (bt[$i] or $b2))**

It is not possible to call a function in expressions.

**Prohibit evaluation by using “”, example:
$cg = “(10,15)”**

8/27/2018 VisionLab 95

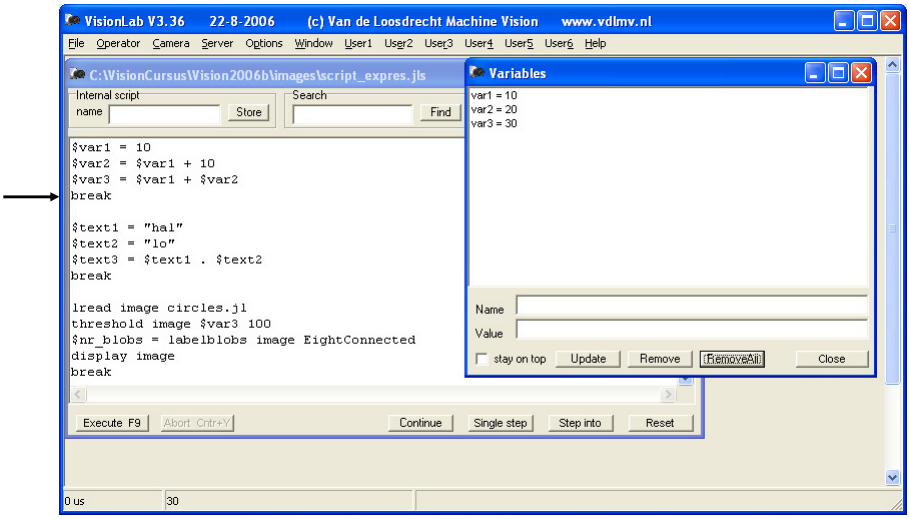
Demonstration script expressions (*)

- **open script script_expres.jls**
- **execute script**
- **open window 'Examine vars' in the server menu**
- **click continue script**
- **click continue script**

- **explain break(points), reset and single stepping scripts**

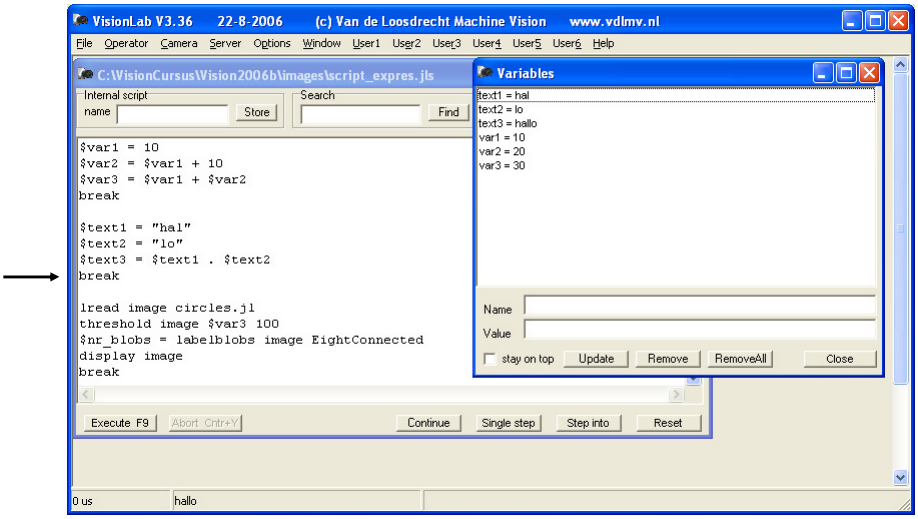
8/27/2018 VisionLab 96

Demonstration script expressions (*)



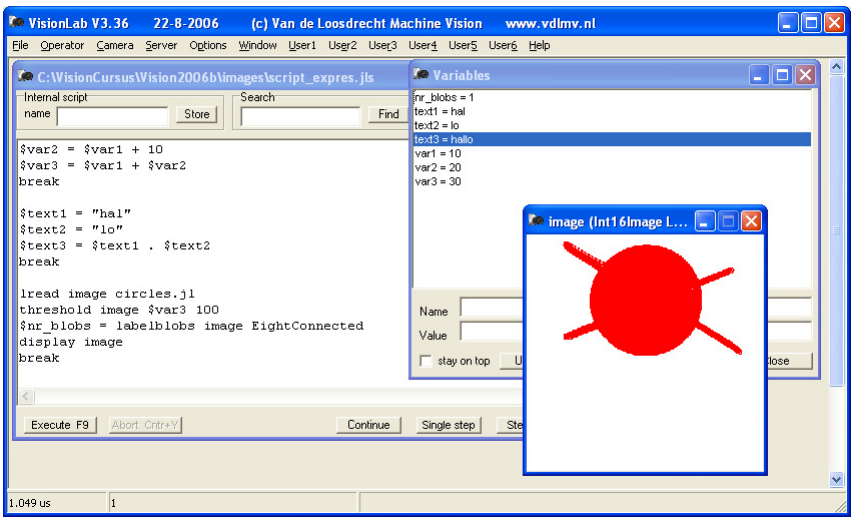
8/27/2018 VisionLab 97

Demonstration script expressions (*)



8/27/2018 VisionLab 98

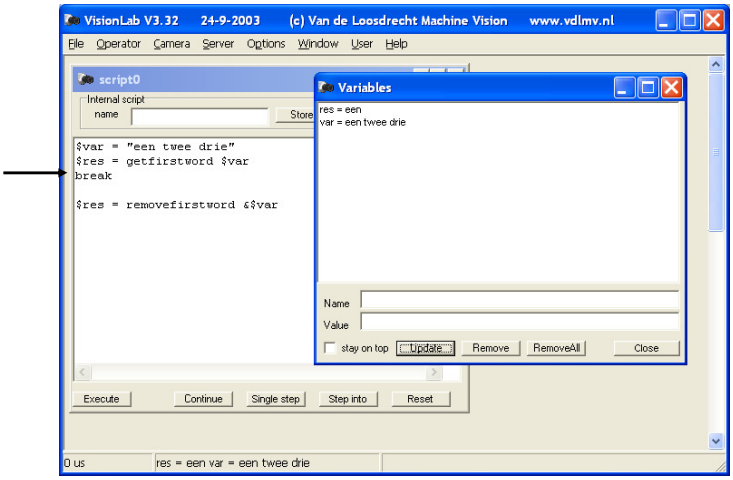
Demonstration script expressions (*)



Operators on variables (*)

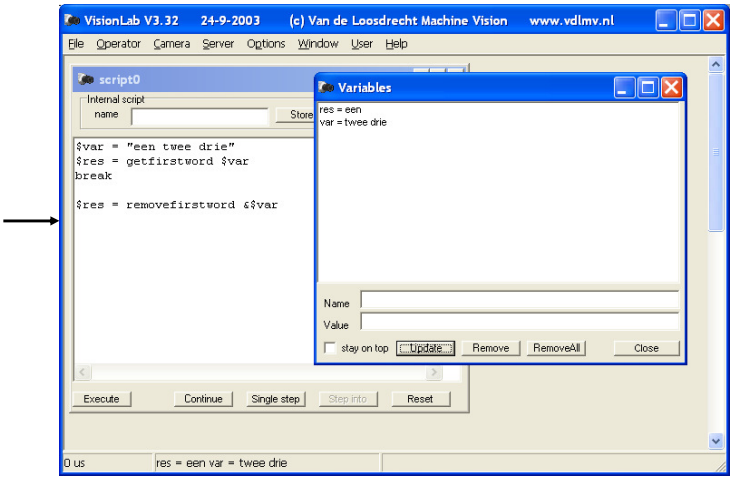
- **Operators:**
  - **for string manipulation**
  - **math operators for number manipulation**
- **Parameter types:**
  - **by value, <var>**
  - **by reference, <&var>**
- **Examples**
  - **by value: $res = getfirstword $var**
  - **by reference: $res = removefirstword &$var**

Parameter by value (*)



8/27/2018 VisionLab 101

Parameter by reference (*)



8/27/2018 VisionLab 102

Operators for string manipulation (*)

A variable can be initialized with a string using double quotes.
Example: $s = "here are spaces allowed"

. is dyadic operator used for string concatenation.

Note this operator is used for concatenation of chars in a word.
If used as $v . $w, both $v and $w must be non empty variables with one word as value.
Use for concatenation of words the string operator Concat.

8/27/2018 VisionLab 103

Operators for string manipulation (*)

A variable can be initialised with a string using double quotes.
Example: $s = "here are spaces allowed"

- **getfirstword <var>**
- **getfirstnwords <n> <var>**
- **getnthword <n> <var>**
- **getnthfromvector <n> <var>**
- **getsizeofvector <var>**
- **vector <i1> <i2> <i3> .....**
- **listallvars**
- **removefirstword <&var>**
- **removefirstnwords <n> <&var>**
- **removeuntilword <word> <&var>**
- **strip <&var>**

8/27/2018 VisionLab 104

Operators for special string manipulation (*)

- **getvar <&var>**
- **isequalvar <&var1> <&var2>**
- **removevar <&var>**
- **removeallvars**
- **setvar <&var> <value>**
- **isvar <varname without $>**
- **testequalvar <&var1> <&var2>**
- **concat <sequence of <var> and text>**
- **substr <&var> <offset> <count>**
- **length <&var>**
- **find <&str> <key> <offset>, find substring in string**
- **findfirstof <&str> <key> <offset>, find character in string**
- **findlastof <&str> <key> <offset >, find character in string**

For more details of operators, see online help

8/27/2018 VisionLab 105

Math operators for vars (*)

- **asin <x>**
- **acos <x>**
- **atan <x>**
- **atan2 <y> <x>**
- **ceil <x>**
- **cos <x>**
- **cosh <x>**
- **exp <x>**
- **fabs <x>**
- **floattoint <x>**
- **floor <x>**
- **fmod <x> <y>**
- **initrandomgen <x>**
- **log <x>**
- **log10 <x>**
- **mod <x> <y>**
- **polarlinesintersection <pc1> <pc2>**
- **pow <x> <y>**
- **random <low> <high>**
- **randomint <low><high>**
- **sgn<x>**
- **sin <x>**
- **sinh <x>**
- **sqrt <x>**
- **tan <x>**
- **tanh <x>**

8/27/2018 VisionLab 106

Script control statements (*)

Nesting to arbitrary depth is possible

```
if <condition> then
   <statement block>
else
  <statement block>
endif

while <condition> do
   <statement block>
endwhile

 for<var> = <expression> (<to>|<downto>) <expression> do
    <statement block>
 endfor
```


Script control statements (*)

```
switch <expression>
case <tag>
   <statement block>
case <tag>
   <statement block>
default
   <statement block>
endswitch

label <labelName>
goto <labelName>
```

Script control statements (*)

return <expression> // function return

notes about string result conventions in VisionLab:
- in case of errors string starts with "[<name of command>]",
so '[' and ']' can NOT be used in normal string results
- "{xxx us} is append for timing information,
so '{' and '}' can NOT be used in normal string results

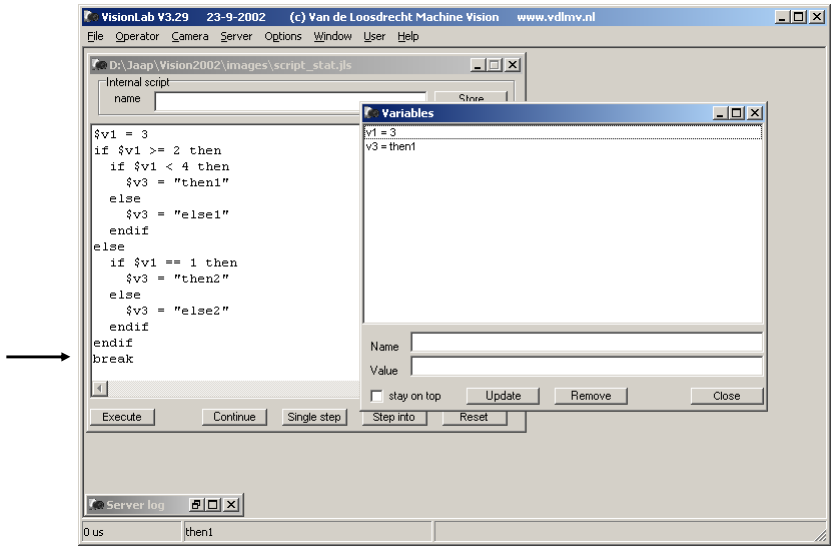
8/27/2018 VisionLab 109

Demonstration script control statements (*)

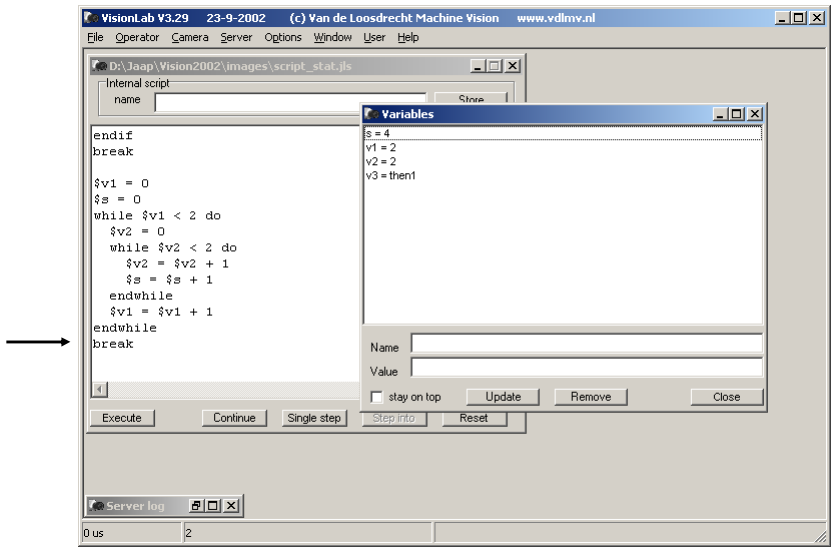
- **open script script_stat.jls**
- **execute script**
- **open window 'Examine vars' in the server menu**
- **click continue script**
- **click continue script**

8/27/2018 VisionLab 110

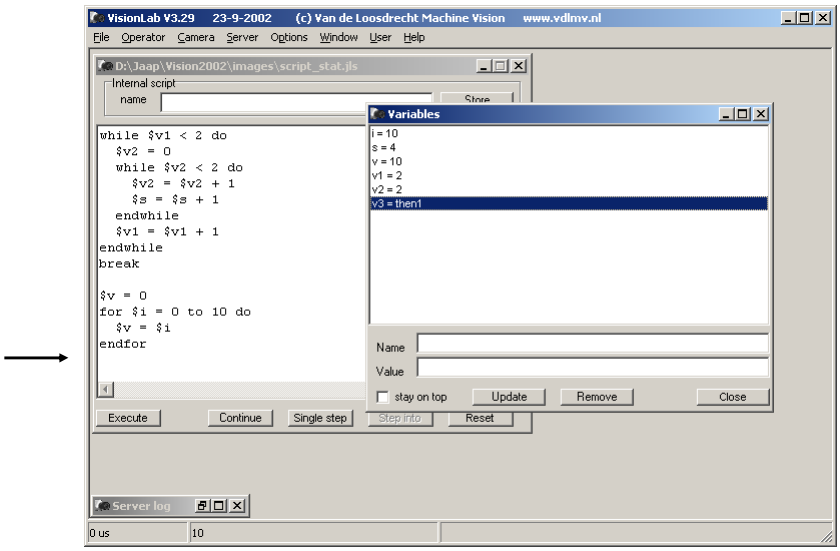
Demonstration script control statements (*)



Demonstration script control statements (*)



Demonstration script control statements (*)



Internal commands (*)

Internal commands are special local commands for which there is not a (direct) server command.

Examples :

- **lcwd <path>: change working directory of client to path**
- **lpwd: return working directory of client.**
- **lread <imagename> <filename>: The file with <filename> is read on the client and added to the server as image with <imagename>**
- **lwrite <imagename> <filename>: The image with <imagename> is written on the client as file with <filename>**
- **display <name>: The image with <name> is displayed**
- **GUIForm: create dialog box from script**
- **see also on-line help for more examples**

Demonstration GUIForm: create dialog box from script (*)

GUIForm <caption> {<spinedit> | <editbox>}

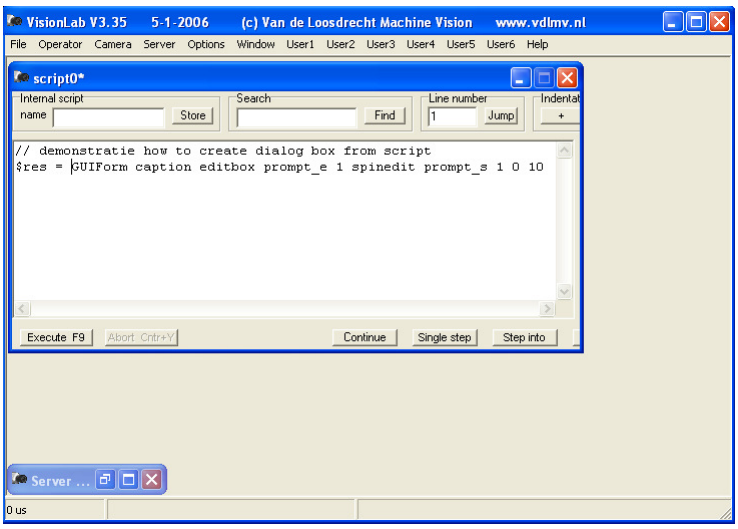
Displays a dynamic form from a script in order to query the user with inputs, result is a string with the answers.

- spinedit: <prompt> <default> <low> <high>
A spinedit field is added to the form with a prompt and a default value. Low and high specify the extreme possible values.

- editbox: <prompt> <default>
An editfield is added to the form with a prompt and a default value.

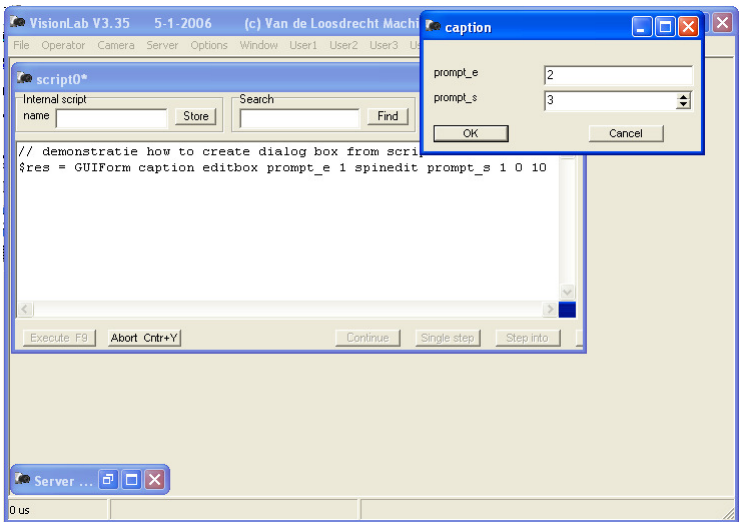
8/27/2018 VisionLab 115

Demonstration GUIForm: create dialog box from script (*)



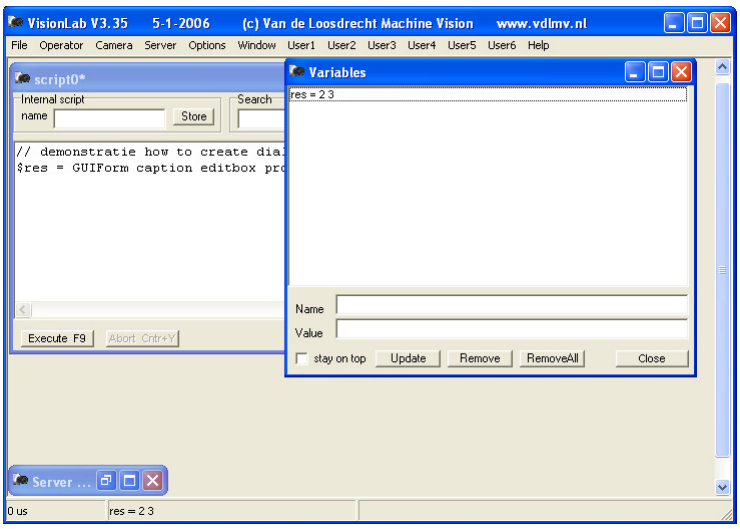
8/27/2018 VisionLab 116

Demonstration GUIForm: execute script (*)



8/27/2018 VisionLab 117

Demonstration GUIForm: examine result variable (*)



8/27/2018 VisionLab 118

Special server commands (*)

Examples :

- **cwd <path>: change working directory of server to path**
- **pwd: return working directory of server**
- **read <imagename> <filename>: The file with <filename> is read on the server and added to the server as image with <imagename>**
- **write <imagename> <filename>: The image with <imagename> is written on the server as file with <filename>**
- **Date and time command, see next slide**
- **see also on-line help for more examples**

8/27/2018 VisionLab 119

Date and time commands (*)

- **Time: returns <day> <month> <dayInMonth> <hh:mm:ss> <year>**

- **Delay <secs>: delay operations for secs seconds time.**
- **MicroDelay <us>: delay operations for us micro-seconds time.**
- **MilliDelay <ms>: delay operations for ms milli-seconds time.**

- **ResetTimeStamp: reset time stamp counter to zero**
- **MicroTimeStamp: return a time stamp in micro seconds, note: maximum time for overflow after reset is 35 minutes.**
- **MilliTimeStamp: return a time stamp in mille seconds, note: maximum time for overflow after reset is 583 hours.**

8/27/2018 VisionLab 120

Pre-processor commands (*)

A command is pre-processed before it is executed. The following pre-processor commands are only available in the client:

- **%currentimage: this command is replaced with the name of the current selected image or the name of the image selected as script image**
- **%secondimage: this command is replaced with the name of the image selected as second image**
- **%thirdimage: this command is replaced with the name of the image selected as third image**
- **%startupdir: directory name from which the client has been launched**

8/27/2018 VisionLab 121

Script procedure calls (*)

There are three different kind of scripts:

- **local scripts: script is executed in the client or called with the *lcall* command from a script executed in the client**
- **remote file scripts: script is executed in the server. Script is started with a *call* <remote fileName> <params> command**
- **internal script: script is executed in the server. Before being started with a *icall* <internal scriptName> <params> command the script has to be stored in the server**

With the command IsServerScript can be tested whether a script is running on the client side or server side.

Function result is true or false.

8/27/2018 VisionLab 122

Script procedure calls (*)

The formal parameters have the form %pn, where n starts at 1
The first parameter is %p1, the next %p2 etc
Arrays can be passed as actual parameters using &$arrayName.

A script can return a function result with the return command
return <function result>

Local scripts can be aborted with <control y>

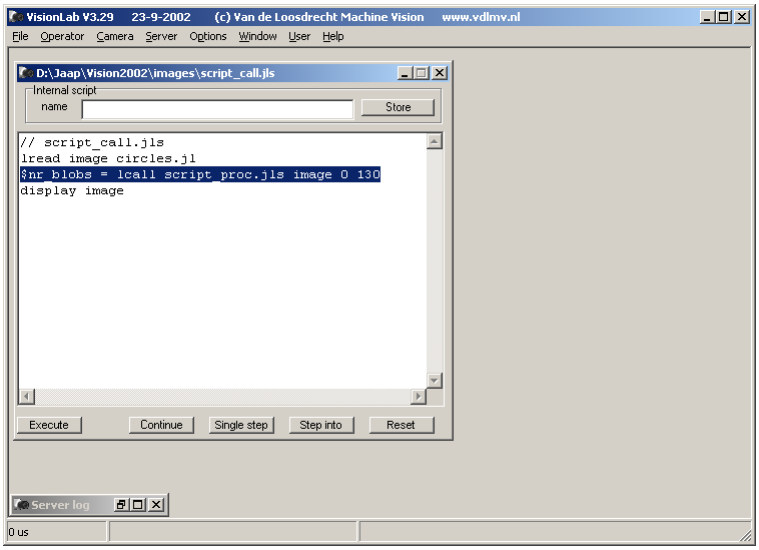
8/27/2018 VisionLab 123

Demonstration script procedure calls (*)

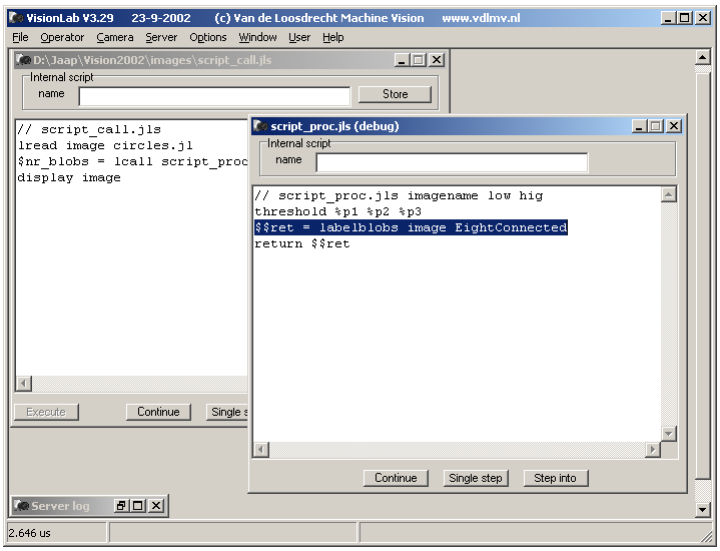
- **open script script_call.jls**
- **click single step**
- **click single step**
- **click step into**
- **click single step**
- **click single step**
- **open window 'Examine vars' in the server menu**
- **click continue script**

8/27/2018 VisionLab 124

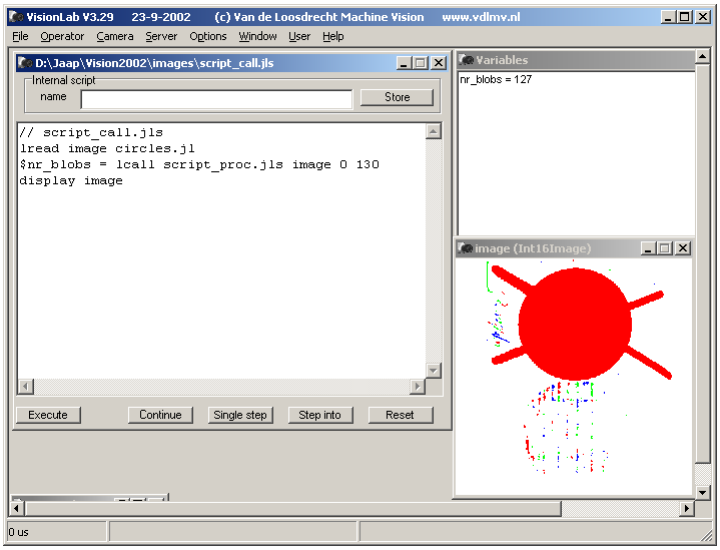
Demonstration script procedure calls (*)



Demonstration script procedure calls (*)



Demonstration script procedure calls (*)



Array as parameters, method 1 (*)

Main:

```
$arr[0] = 1
$arr[1] = 2
lcall arrayParam.jls &$arr
```

arrayParams2.jls:

```
CopyVar %p1 &$res
```

Array as parameters, method 2 (*)

Main:

```
$arr[0] = 1
$arr[1] = 2
lcall arrayParam2.jls arr
```

arrayParams2.jls:

```
CopyVar &$%p1 &$res
```


Speeding up scripts (*)

- **Using internal scripts**
- **System settings**
  - **Disable client logging**
  - **Disable server logging**
  - **Disable history mechanism**

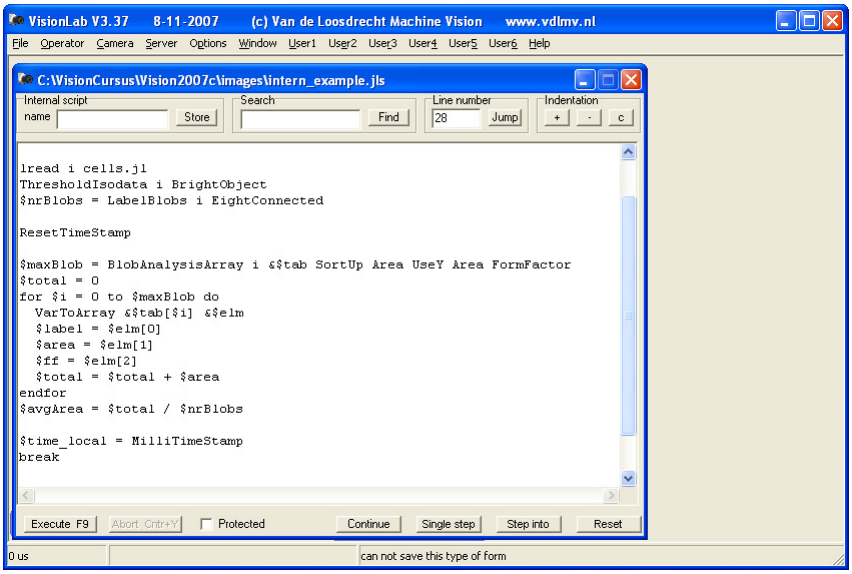
Demonstration speed up by using internal script (*)

- open script internal_example.jls
- run script to first break point
- examine variables
- examine script intern_script.jls
- run (continue) second part of script and compare executing times
- $time_local = 6411 ms
- $time_intern = 176 ms

- Note: this example is just to demonstrate internal scripts. A much faster way to calculate the average area is to count the number of object pixels and divide this number with the number of object

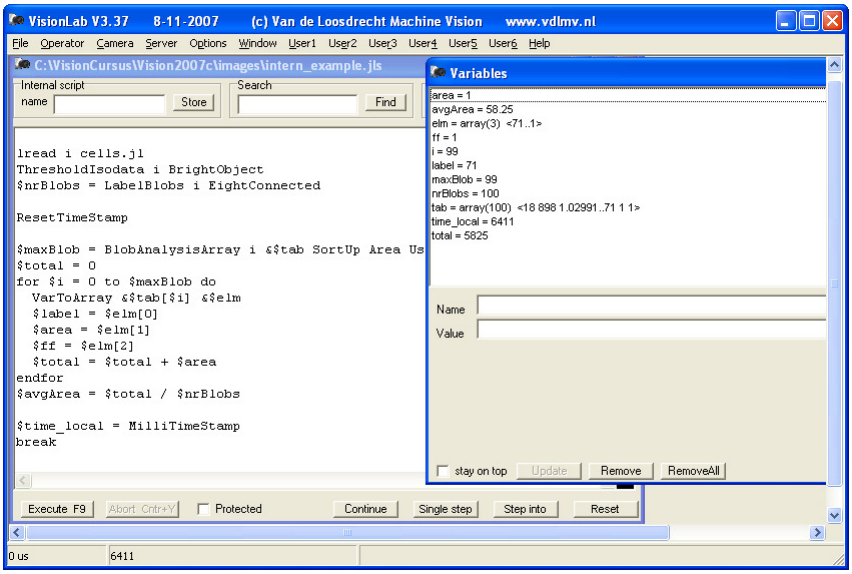
8/27/2018 VisionLab 131

open script internal_example.jls (*)

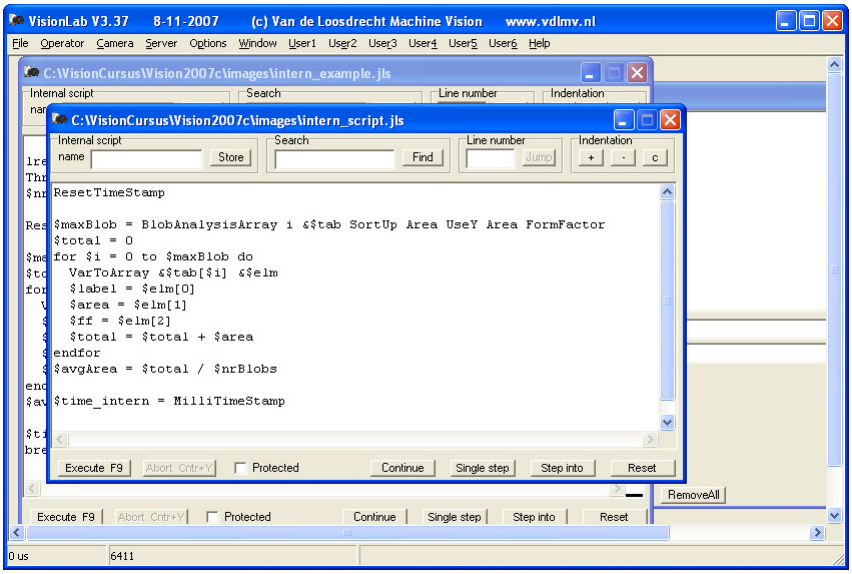


8/27/2018 VisionLab 132

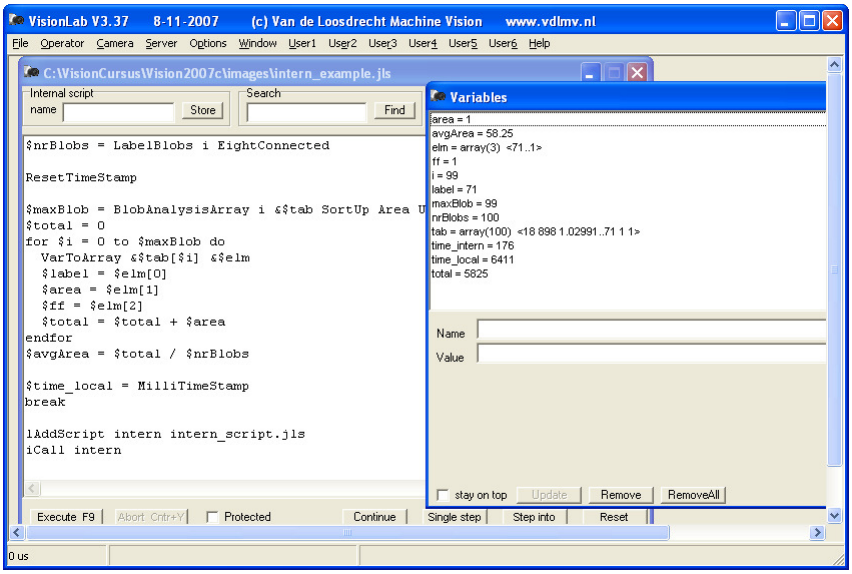
run script to first break point (*)



examine script intern_script.jls (*)



run second part of script and compare executing times(*)



Speeding up scripts (*)

- **System settings**
  - **Disable client logging:**
    set in main menu item server log to "no log"
  - **Disable server logging and Disable history mechanism:**
    close visionlab
    open file visionlab.ini
    search in file for "[localserver]"
    next line has item "exename" with syntax
    vissvr.exe <port> <byteorder> <timeout> <maxhissize> <EchoOn|EchoOff> <(no)debug>
    set <maxhissize> to 0
    set <EchoOn|EchoOff> to EchoOff
    example: "exename=vissvr.exe 2066 NativeByteOrder 5 0 EchoOff nodebug"
    save and close file visionlab.ini
    start VisionLab again

Adding a script as new operator (*)

New operators can be added to the system by editing the file visionlab.ini. See for more details Adding a new operator to the client

Example:
An operator with the name Script must be added to the usermenu. When the operator is selected the script with name test.jls is to be executed and it will receive three parameters.
In order to make this possible add the following lines to the file visionlab.ini and startup the system.

```
[examplescript]
menu=UserMenu
caption=Example &Script
class=IntImage
script=test.jls
paramform=scriptparam
selectedimage source
editbox low 0
editbox high 80
```

8/27/2018 VisionLab 137

Adding a script as new operator (*)

Explanation:

[examplescript]: In [ ] the name of the operator. Note: no white space behind] is allowed.

menu: The menu where the operator is added to the system.
caption: The name used for the operator in the menu. The & give the letter for the shortkey.

class: The highest class in the Image hierarchy overview on which the operator can work.

script: The filename of the script which is to be executed.

paramform: Specifies the form which to be used to ask the user for the parameters. Currently for user scripts only 'scriptparam' is supported.

8/27/2018 VisionLab 138

Adding a script as new operator (*)

The section following paramform specifies the parameters for the operator. The maximal number of parameters is 25. The parameters are displayed and send to the server in the same order as they are declared. There are the following possibilities:

- **selectedimage <prompt>: An edit field is added with a prompt and the image name of the currently selected image**
- **2ndselected <prompt>: An edit field is added with a prompt and the image name of the 2ndselected image**
- **3rdselected <prompt>: An edit field is added with a prompt and the image name of the 3rdselected image.**
- **genimagename <prompt>: An edit field is added with a prompt and a generated image name**
- **spinedit <prompt> <default> <low> <high>: A spinedit field is added with a prompt and a default value. Low and high specify the extreme possible values**

8/27/2018 VisionLab 139

Adding a script as new operator (*)

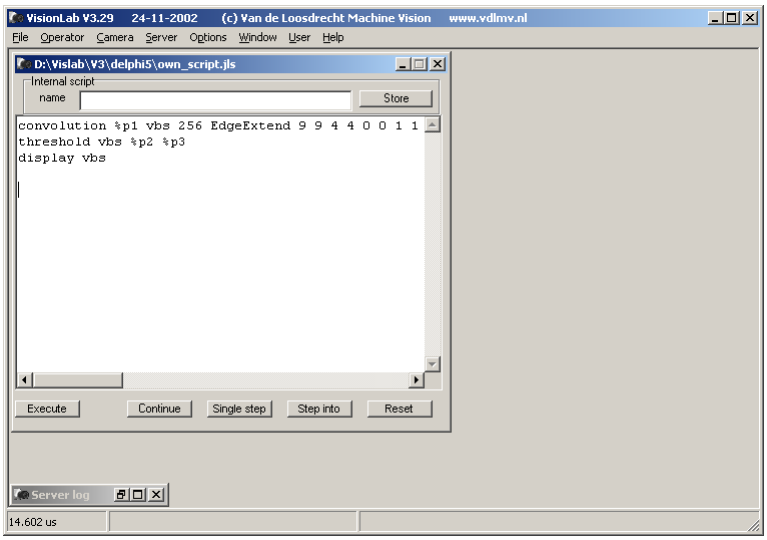
- **editbox <prompt> <default>: An edit field is added with a prompt and a default value**
- **combobox <prompt> <default> <item1> ... <itemN>:**
  A combo box is added with a prompt and a default value. The combobox is filled with the specified items. If prompt equals to 'none' combobox is invisible

8/27/2018 VisionLab 140

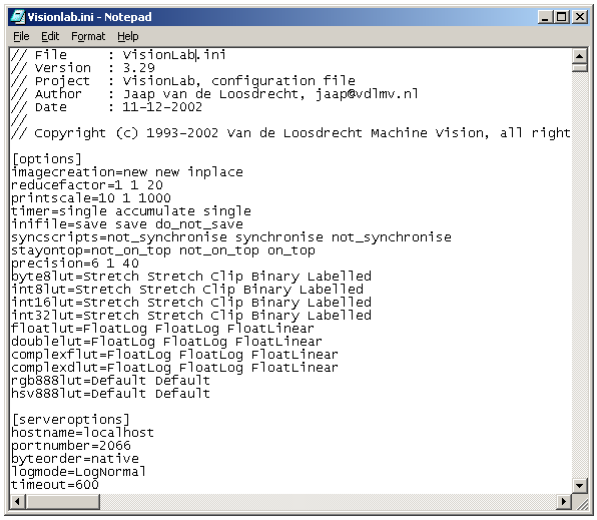
Example: Adding an own script as operator (*)

- **Demonstrate own_script.jls in User Menu item 'example script' on circles.jl and look at vislab.ini for added script as new operator.**

Example: Adding an own script as operator (*)

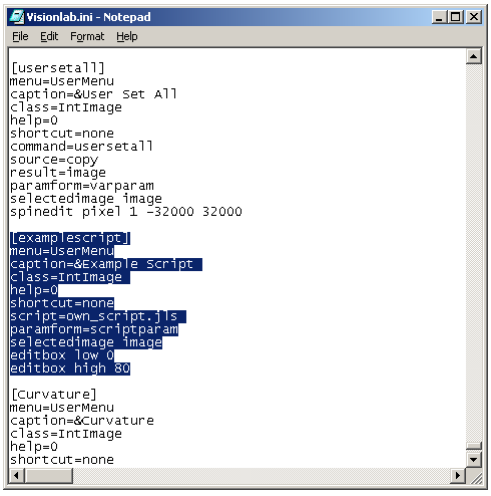


VisionLab.ini configuration file (*)



8/27/2018 VisionLab 143

Adding script own_script.jls as operator in visionlab.ini (*)



8/27/2018 VisionLab 144

Adding a user C++ operator (*)

Add new operator to server to set all pixels to same value
- Write new operator in C++ (UserSetAll)
- Write envelope for operator (SetAllCmd)
- Add envelope to cmd interpreter (Main in visserv.cpp)

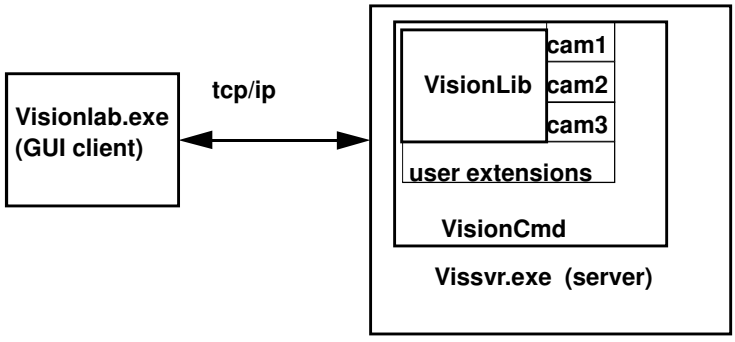
- Command send to server will be: usersetall <image name> <pixel value>
- The server will use the first word as name of the command ("usersetall") and will use this name to find the associated envelope function SetAllCmd
- SetAllCmd will get as input stream the command without the first word: "<image name> <pixel value>"
- SetAllCmd will:
  - read the image name and pixel value from the inputs stream
  - check whether image name is a valid image
  - call UserSetAll with image and value as parameter
  - the call to userSetAll is surrounded with timer functions which supports the displaying of the time needed for the operation in the left bottom corner of the GUI
  - The call AddLastCmdToHistory adds the command to the history list of the image

Add new operator to GUI
- Edit vislab.ini

See for example the files in the src directory of the SDK:
useroper.h, useroper.cpp and visserv.cpp

8/27/2018 VisionLab 145

Adding a C++ operator to the VisionLab server (*)



The new operator is added in the next example in the module useroper.cpp

See for files, the solution vissvr.sln

8/27/2018 VisionLab 146

Useroper.cpp (*)

Example operator to set all pixels to same value

```
// The operator to be added
void UserSetAll (Int16Image &image, const Int16Pixel pixel)  {
  for (int y = 0; y < image.GetHeight(); y++) {
    for (int x = 0; x < image.GetWidth(); x++) {
      image(x,y) = pixel;
    } // for x
  } // for y
} // UserSetAll
```

Example string send to the command interpreter: "usersetall image 100"

```
// the following code envelope is necessary to add the new operator to the server
static void SetAllCmd (istream &is, ostream &os, VisLibCmdInt &cmdInt) {
  QString imageName;
  Int16Pixel pixel;
  is >> imageName >> pixel;
  if (is.fail()) throw(Error("[SetAll] input error"));
  if (CheckIsInt16Image(cmdInt, imageName, os)) {
    Int16Image *src = dynamic_cast <Int16Image *>(cmdInt.GetImage(imageName));
    cmdInt.StartTimer();
    UserSetAll (*src, pixel);
    cmdInt.StopTimer();
    cmdInt.AddLastCmdToHistory (imageName);
    os << "example of a result string";
  }
} // SetAllCmd

// called by the main program
void InsertUserOperCmds (VisLibCmdInt &cmdInt) {
  cmdInt.InsertCmd ("usersetall",  CmdIntCommand(SetAllCmd, cmdInt), "<imageName> <pixelvalue>");
} // InsertUserOperCmds
```


Vissvr.cpp (*)

```
...
int main (int argc, char *argv[ ]) {
    ...
    VisLibCmdInt cmdInt ( ... );
    VisLibSvr server(cmdInt, ... );
    // register camera's
    RegisterDummyCam(cmdInt);
    //RegisterFirePackCam(cmdInt);
    //RegisterTeliCam(cmdInt);
    //RegisterFireCam(cmdInt);
    //RegisterSKCam(cmdInt);
    //RegisterGCam(cmdInt);
    RegisterDXCam(cmdInt);
    InsertUserOperCmds (cmdInt);   // simple user defined operator
    server.Run();
    ....
} // main
```

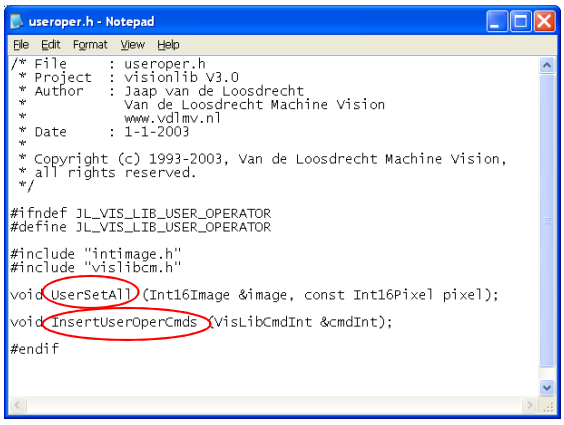
Vissvr.cpp (*)

You need to adapt the source if you want to use an another camera interface

```
/* How to add camera support to this component:
 * DXCam:
 * - add "RegisterDXCam(*cmdInt);" to DllMain
 * - add dxcam.lib from the ReleaseLib directory to this project
 * - build the project
 * Firewire FirePack:
 * - add "RegisterFirePackCam(*cmdInt);" to main
 * - add firepack.lib from the ReleaseLib directory to this project
 * - build the project
 * Gen<I>Cam with GenTL:
 * - add "RegisterJL_GenICam(*cmdInt);" to main
 * - add genicam.lib from the ReleaseLib directory to this project
 * - build the project
 */
```

8/27/2018 VisionLab 149

useroper.h (*)



8/27/2018 VisionLab 150

useroper.cpp (*)

useroper.cpp - Notepad

File Edit Format View Help

```
using namespace JL_VisionLib_V3;

// the new operator itself

void UserSetAll (Int16Image &image, const Int16Pixel pixel) {
   for (int y = 0; y < image.GetHeight()  ; y++) {
      for (int x = 0; x < image.GetWidth()  ; x++) {
         image(x,y) = pixel;
      } // for x
   } // for y
   /* alternative faster solution using pointers:
   Int16Pixel *ownPtr = image.GetFirstPixelPtr();
   Int16Pixel *lastPtr = image.GetLastPixelPtr();
   while (ownPtr <= lastPtr) {
      *ownPtr++ = pixel;
   } // for all pixels */
} // UserSetAll

// the following code envelope is necessary to add the new operator to the server

static void SetAllCmd (istream &is, ostream &os, VisLibCmdInt &cmdInt) {
   QString imageName;
   Int16Pixel pixel;
   is >> imageName >> pixel;
   if (is.fail()) throw(Error("[SetAll] input error"));
   if (checkIsInt16Image(cmdInt, imageName, os)) {
      Int16Image *src = dynamic_cast <Int16Image *>(cmdInt.GetImage(imageName));
      cmdInt.StartTimer();
      UserSetAll (*src, pixel);
      cmdInt.StopTimer();
      cmdInt.AddLastCmdToHistory (imageName);
      os << "example of a result string";
   }
} // SetAllCmd

// called by the main program
void InsertUserOperCmds (VisLibCmdInt &cmdInt) {
   cmdInt.InsertCmd ("UserSetAll",  CmdIntCommand(SetAllCmd, cmdInt), "<imageName> <pixelvalue>");
} // InsertUserOperCmds
```

the operator to add

the envelope function

to be called from main

visserv.cpp (*)

visserv.cpp - Notepad

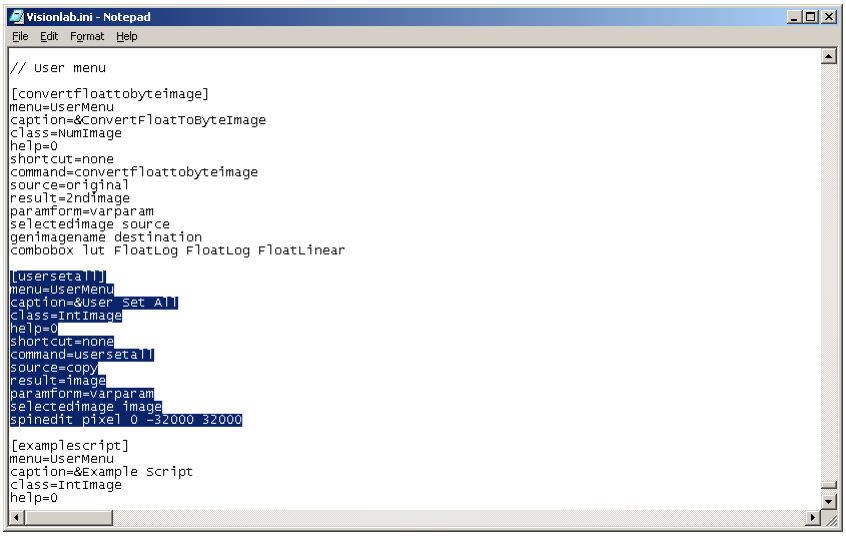
File Edit Format View Help

```
.....
int main (int argc, char *argv[])
   .....

      VisLibCmdInt cmdInt(debug);
      VisLibSvr server(cmdInt, port, byteOrder, timeOut);

      // register camera's
      // RegisterFireCam(cmdInt);
      RegisterDummyCam(cmdInt);
      RegisterFileCam(cmdInt);
      //RegisterRioCam (cmdInt);
      // examples:
      InsertUserOperCmds (cmdInt);   // simple user defined operator
      // insert your own cmds <----------
      server.Run();
   }
   ....
} // main
```

Add new operator to GUI (*)



8/27/2018 VisionLab 153

Adding a new C++ operator to the client (*)

New operators can be added to the client by editing the file visionlab.ini. The new operators will be active the next time the system is started.

The descriptor for the operator with name usersetall looks like:

```
[usersetall]
menu=User1Menu
caption=&User Set All
class=IntImage
help=0
shortcut=none
command=UserSetAll
source=copy
result=image
paramform=varparam
selectedimage image
spinedit pixel 1 -32000 32000
```

8/27/2018 VisionLab 154

Adding a new C++ operator to the client (*)

[usersetall]: In [ ] the name of the operator. Note: no white space behind] is allowed.

menu: The menu where the operator is added to the system.

caption: The name used for the operator in the menu. The & give the letter for the shortkey.

class: The highest class in the Image hierarchy overview on which the operator can work, or "NoImage" if operator does not work with images.

shortcut: The name for a key on the keyboard, which can activate the operator without using the menu. Possible values: none, ctrl+a, .. ctrl+z, f1, .. f12, ctrl+f1, .. ctrl+f12, shift+f1, .. shift+f12, shift+ctrl+f1, .. shift+ctrl+f12, ins, shift+ins, ctrl+ins, del, shift+del, ctrl+del, alt+bksp and shift+alt+bksp.

command: The command name which is sent to the server.

8/27/2018 VisionLab 155

Adding a new C++ operator to the client (*)

source: Specifies on which image operation is performed, possible values:

- **original: operation is performed on the source image.**
- **copy: operation is performed on a copy of the original, the name of the copy is either specified by a imagename parameter (see below: description of paramform) or generated by the system.**
- **none: operation is not performed on an image.**

result: Determines how the result of the operation is to be displayed, possible values:

- **image: The image (= first parameter) on which the operation was performed is displayed.**
- **2ndimage: The image specified as second parameter in the parameter list of the server command is displayed.**
- **3rdimage: The image specified as third parameter in the parameter list of the server command is displayed.**
- **4thimage: The image specified as fourthparameter in the parameter list of the server command is displayed.**
- **2nd3rdimage: The images specified as second and third parameter in the parameter list of the server command are displayed.**

8/27/2018 VisionLab 156

Adding a new C++ operator to the client (*)

result (continued):

- **2nd3rd4thimage: The images specified as second, third and fourth parameter in the parameter list of the server command are displayed.**
- **bothimages: image and 2ndimage are displayed, see above.**
- **box: As result a string with the format '(x1,y1) (x2,y2)' is expected from the server. A box with lefttop (x1,y1) and rightbottom (x2,y2) is drawn in the current active image.**
- **circle: As result a string with the format '(x,y) r h' is expected from the server. A circle with centre (x,y) and radius r is drawn in the current active image. h is the number of hits.**
- **circles: As result a string with the format 'n (x,y) r h ... (x,y) r h' is expected from the server. N circles with centres (x,y) and radius r are drawn in the current active image.**
- **line: As result a string with the format '(x1,y1) (x2,y2) is expected from the server. A line from (x1,y1) to (x2,y2) is drawn in the current active image.**
- **polarline: As result a string with the format '(r,phi) h' is expected from the server. A line with polarcoordinates is drawn in the current active image. h is the number of hits.**

8/27/2018 VisionLab 157

Adding a new C++ operator to the client (*)

result (continued):

- **polarlines: As result a string with the format 'n (r,phi) h ... (r,phi) h is expected from the server. N lines with polarcoordinates is drawn in the current active image. h is the number of hits.**
- **string: As result a string is expected from the server which is displayed in the info bar.**
- **stringlist: As result a list of strings is expected from the server which is displayed in a window.**
- **nothing: Nothing is displayed.**

paramform: Specifies the form which to be used to ask the user for the parameters. Currently for user C++ operators only 'varparam' is supported.

The section following paramform specifies the parameters for the operator. The maximal number of parameters is 25. The parameters are displayed and sent to the server in the same order as they are declared.

8/27/2018 VisionLab 158

Adding a new C++ operator to the client (*)

There are the following possibilities for the parameters :

- **selectedimage <prompt>: An editfield is added with a prompt and the image name of the currently selected image.**
- **2ndselected <prompt>: An editfield is added with a prompt and the image name of the 2ndselected image.**
- **3rdselected <prompt>: An editfield is added with a prompt and the image name of the 3rdselected image.**
- **genimagename <prompt>: An editfield is added with a prompt and a generated image name.**
- **spinedit <prompt> <default> <low> <high>: A spinedit field is added with a prompt and a default value. Low and high specify the extreme possible values.**
- **editbox <prompt> <default>: An editfield is added with a prompt and a default value.**
- **combobox <prompt> <default> <item1> ... <itemN>: A combo box is added with a prompt and a default value. The combobox is filled with the specified items.**

8/27/2018 VisionLab 159

Adding a new C++ operator to the client (*)

Possibilities for the parameters (continued):

- **checklistbox <prompt> <defaults> <|> <item1> ... <itemN>: A checklistbox is added with a prompt and default check values. The listbox is filled with the specified items.**
  NOTE: this parameter can ONLY be used as the last parameter of a command.

8/27/2018 VisionLab 160

Adding a user plugin with new operators (*)

Steps

- **Write plugin with new operators**
- **Add for each new operator a line to file pluginsJL.ini**
- **Add for each operator a description in visionlab.ini (see slide about "Adding a new C++ operator to the client"**

See for examples the following files in the SDK:

- **Header file for plugin interface: src\JL_VisionLib_plugin.h**
- **Example solution for plugin for C++: msvs08\Examples\JL_VisionLib_plugin**
- **Example project plugin for Delphi: msvs08\Examples\TestDelphi5**

8/27/2018 VisionLab 161

JL_VisionLib_plugin.h (*)

This header file defines the interface for the PlugIn.

It consists of following parts:

- **Enum JL_ImageType for defininig the image types of VisionLab and BytesPerPixelTab table for defining the size of a pixel for each image type**
- **Prototypes for the callback functions**
- **JL_PlugInInitialize, this function is called to initialize the callback functions when linking the DLL**
- **The prototype for the plugin functions inside the DLL**

Notes:

- **there can be any number of plugin functions in one dll**
- **there can be any number of dlls**

8/27/2018 VisionLab 162

JL_VisionLib_plugin.h (*)

```
//Image type enumeration
enum JL_ImageType {BYTEIMAGE, INT8IMAGE, INT16IMAGE, INT32IMAGE, FLOATIMAGE,
    DOUBLEIMAGE, RGB888IMAGE, RGB161616IMAGE, HSV888IMAGE, HSV161616IMAGE,
    YUV888IMAGE, YUV161616IMAGE, COMPLEXFLOATIMAGE, COMPLEXDOUBLEIMAGE
};

// table with the bytes per pixel for each image type
const int JL_BytesPerPixelTab[] = {
   sizeof(unsigned char),
    sizeof(char),
    sizeof(short int),
    sizeof(int),
    sizeof(float),
    sizeof(double),
    sizeof(unsigned char) * 4,
    sizeof(unsigned short int) * 3,
    sizeof(unsigned char) * 4,
    sizeof(unsigned short int) * 3,
    sizeof(unsigned char) * 4,
    sizeof(unsigned short int) * 3,
    sizeof(float) * 2,
    sizeof(double) * 2
};
```

8/27/2018 VisionLab 163

JL_VisionLib_plugin.h (*)

```
//Callback to create a new image
typedef void * (*JL_CreateImageProc) (const char *imageName,
               const JL_ImageType type, int height, int width, void *cmdInt);

//Callback to get image information
typedef void * (*JL_GetImageProc) (const char *imageName, JL_ImageType *type,
               int *height, int *width, void *cmdInt);

//Callback to create a new string
typedef char * (*JL_CreateStringProc) (const int length);

// Callback to execute VisionLab command from plugin, plugin must reserve memory
    with size maxSizeResult for result
typedef void (*JL_ExecuteCmdProc) (const char *cmd, char * const result,
             const int maxSizeResult, void *cmdInt);
```

8/27/2018 VisionLab 164

JL_VisionLib_plugin.h (*)

```
extern "C" __declspec(dllexport)
//This function is called to initialize the callback functions when linking the DLL
void JL_PlugInInitialize (JL_CreateImageProc createImageProc,
                          JL_GetImageProc getImageProc,
                          JL_CreateStringProc createStringProc,
                          JL_ExecuteCmdProc executeCmdProc,
                          void *cmdInt);

/*
The prototype for the plugin functions inside the DLL
#ifdef WIN32
extern "C" __declspec(dllexport)
#endif
int JL_PlugIn (const int nrParams, const char* const *paramArray, char** result, void
    *cmdInt);
function result == 0 means success
function result != 0 means failure, possible extra error information in result
notes about string result conventions in VisionLab:
- in case of errors string starts with "[<name of command>]",
  so '[' and ']' can NOT be used in normal string results
- "{xxx us} is append for timing information,
  so '{' and '}' can NOT be used in normal string results
*/
```


Example in C++ for plugin: JL_VisionLib_plugin.cpp (*)

This example has the following parts:

- **Implementation of JL_PlugInInitialize**
- **JL_Echo_PlugIn**
  example of string handling
- **JL_CmdTest_PlugIn**
  example of calling a VisionLab command from the plugin
- **JL_CopyImage_PlugIn**
  example of making a copy of an image
- **JL_Threshold_PlugIn**
  example of a user threshold implementation

JL_VisionLib_plugin.cpp: Implementation of JL_PlugInInitialize (*)

```
/* Start of code that should always be in the plugin */
static void * globalCmdInt;
static JL_CreateImageProc createImageProc;
static JL_GetImageProc getImageProc;
static JL_CreateStringProc createStringProc;
static JL_ExecuteCmdProc executeCmdProc;
// This function is called when linking the DLL
void JL_PlugInInitialize (JL_CreateImageProc createImageP,
                          JL_GetImageProc getImageP,
                          JL_CreateStringProc createStringP,
                          JL_ExecuteCmdProc execP,
                          void *cmdInt) {
   createImageProc = createImageP;
   getImageProc = getImageP;
   createStringProc = createStringP;
   executeCmdProc = execP;
   globalCmdInt = cmdInt;
} // JL_PlugInInitialize
/* End of code that should always be in the plugin */
```

8/27/2018 VisionLab 167

JL_VisionLib_plugin.cpp: JL_Echo_PlugIn (*)

```
// Example of handling strings
// - input string is copied to function result.
// - $res = Echo_PlugIn 12345
// - $res will have the value "12345"
extern "C" __declspec(dllexport)
int JL_Echo_PlugIn (const int nrParams, const char* const *paramArray,
                    char** result, void *cmdInt) {
   if (nrParams == 1) {
      *result = createStringProc ((unsigned int)strlen(paramArray[0]));
      strcpy(*result, paramArray[0]);
   } else {
      string msg = "[JL_Echo_PlugIn]: Invalid number of parameters";
      *result = createStringProc ((unsigned int) msg.size());
      strcpy(*result, msg.c_str());
      return -1; // indicating failure
   } // if
   return 0;  // indication succes
} // JL_Echo_PlugIn
```

8/27/2018 VisionLab 168

JL_VisionLib_plugin.cpp: JL_CmdTest_PlugIn (*)

```
// Example of executing VisionLab command
// - 1 input parameter
// - command to execute:  $test_plugin = 'value first input parameter'
// - no result is returned
extern "C" __declspec(dllexport)
int JL_CmdTest_PlugIn (const int nrParams, const char* const *paramArray,
                       char** result, void *cmdInt) {
   const int bufsize = 100;
   char buf[bufsize];
   if (nrParams == 1) {
      *result = createStringProc (bufsize);
      string cmd =  string("$test_plugin = ") +  string(paramArray[0]);
      executeCmdProc (cmd.c_str(), buf, bufsize, cmdInt);
      strcpy(*result, buf);
   } else {
      string msg = "[JL_CmdTest_PlugIn]: Invalid number of parameters";
      *result = createStringProc ((unsigned int) msg.size());
      strcpy(*result, msg.c_str());
      return -1; // indicating failure
   } // if
   return 0;  // indication succes
} // JL_CmdTest_PlugIn
```

8/27/2018 VisionLab 169

JL_VisionLib_plugin.cpp: JL_CopyImage_PlugIn 1 (*)

```
// Example of handling image
// - image with name in first parameter is copied to image with name
//   in second parameter
// - memory for second image is allocated
// - CopyImage_PlugIn <src> <dest>
extern "C" __declspec(dllexport)
int JL_CopyImage_PlugIn (const int nrParams, const char* const *paramArray,
   char** result, void *cmdInt) {
   if (nrParams == 2) {
        JL_ImageType type;
        int height, width;
        void *srcPtr, *destPtr;
        srcPtr = getImageProc(paramArray[0], &type, &height, &width,
                              globalCmdInt);
        if (srcPtr != NULL)
                destPtr = createImageProc (paramArray[1], type, height,
                                           width, globalCmdInt);
        if (srcPtr != NULL && destPtr != NULL) {
        memcpy((void *) destPtr, (void *) srcPtr, height * width *
               JL_BytesPerPixelTab[type]);
```

8/27/2018 VisionLab 170

JL_VisionLib_plugin.cpp : JL_CopyImage_PlugIn 2 (*)

```
        } else {
           string msg = "[JL_CopyImage_PlugIn]: Invalid source
                        or destination image";
           *result = createStringProc ((unsigned int) msg.size());
           strcpy(*result, msg.c_str());
           return -1; // indicating failure
        }
    } else {
       string msg = "[JL_CopyImage_PlugIn]: Invalid number of
                     parameters";
       *result = createStringProc ((unsigned int) msg.size());
       strcpy(*result, msg.c_str());
       return -1; // indicating failure
    } // if
    return 0;  // indication succes
} // JL_CopyImage_PlugIn
```

8/27/2018 VisionLab 171

JL_VisionLib_plugin.cpp : JL_Threshold_PlugIn 1 (*)

```
static void Threshold (const void *imgPtr, const int height,
                       const int width, const JL_ImageType type,
                       const double low, const double high) {
    switch (type) {
         // …… //
         case BYTEIMAGE: {
             unsigned char *ptr = (unsigned char *) imgPtr;
             for (int i = 0; i < height * width; i++) {
                 if (*ptr > low && *ptr <= high)
                     *ptr++ = 1;
                 else
                     *ptr++ = 0;
              } // for i
              break;
         }
         case INT16IMAGE: {
         // …… //
    } // switch type
} // Threshold
```

8/27/2018 VisionLab 172

JL_VisionLib_plugin.cpp : JL_Threshold_PlugIn 2 (*)

```
// Example of inplace image operation
// - image with name in first parameter is thresholded with 2nd
//   parameter as low and 3rd as high
// - Threshold_PlugIn <image> <low> <high>
extern "C" __declspec(dllexport)
int JL_Threshold_PlugIn (const int nrParams, const char* const
                         *paramArray, char** result, void *cmdInt) {
   if (nrParams == 3) {
        int height, width;
        JL_ImageType type;
        void *imgPtr;
        imgPtr = getImageProc(paramArray[0], &type, &height, &width,
                              globalCmdInt);
        if (imgPtr != NULL) {
            double low = atof(paramArray[1]);
            double high = atof(paramArray[2]);
            Threshold(imgPtr, height, width, type, low, high);
```


JL_VisionLib_plugin.cpp : JL_Threshold_PlugIn 3 (*)

```
        } else {
           string msg = "[JL_ThresholdPlugIn]: Invalid source or
                        destination image";
           *result = createStringProc ((unsigned int) msg.size());
           strcpy(*result, msg.c_str());
           return -1; // indicating failure
        }
   } else {
      string msg = "[JL_ThresholdPlugIn]: Invalid number of
                   parameters";
      *result = createStringProc ((unsigned int) msg.size());
      strcpy(*result, msg.c_str());
      return -1; // indicating failure
   } // if
   return 0;  // indication succes
} // JL_ThresholdPlugIn
```

pluginsJL.ini (*)

The file pluginsJL.ini bind command names to entry points in dlls. This file should be in the same directory as vissvr.exe

Each line has the following syntax:

<cmdName> <dllName> <entryName> <help info>

// <space> <comment>

example:

```
Echo_PlugIn  JL_VisionLib_plugin.dll JL_Echo_PlugIn <string_to_echo>
CmdTest_PlugIn JL_VisionLib_plugin.dll JL_CmdTest_PlugIn <param_for_assignment>
CopyImage_PlugIn JL_VisionLib_plugin.dll JL_CopyImage_PlugIn <srcImage> <destImage>
// comment
Threshold_PlugIn JL_VisionLib_plugin.dll JL_Threshold_PlugIn <imageName> <low> <high>
```

8/27/2018 VisionLab 175

Add for each pugin operator a description in visionlab.ini 1 (*)

```
[echo_plugin]
menu=User1Menu
caption=&Echo_PlugIn
class=NoImage
help=0
shortcut=none
command=Echo_PlugIn
source=none
result=string
paramform=varparam
editbox string 12345

[cmdtest_plugin]
menu=User1Menu
caption=&CmdTest_PlugIn
class=NoImage
help=0
shortcut=none
command=CmdTest_PlugIn
source=none
result=string
paramform=varparam
editbox value 12345
```

8/27/2018 VisionLab 176

Add for each pugin operator a description in visionlab.ini 2 (*)

```
[copyimage_plugin]
menu=User1Menu
caption=&CopyImage_PlugIn
class=NumImage
help=0
shortcut=none
command=CopyImage_PlugIn
source=original
result=2ndimage
paramform=varparam
selectedimage source
genimagename destination

[threshold_plugin]
menu=User1Menu
caption=&Threshold_PlugIn
class=OrdImage
help=0
shortcut=none
command=Threshold_PlugIn
source=copy
result=image
paramform=varparam
selectedimage source
editbox low 160
editbox high 255
```

8/27/2018 VisionLab 177

Using VisionLab command interpreter in #C (*)

Overview:

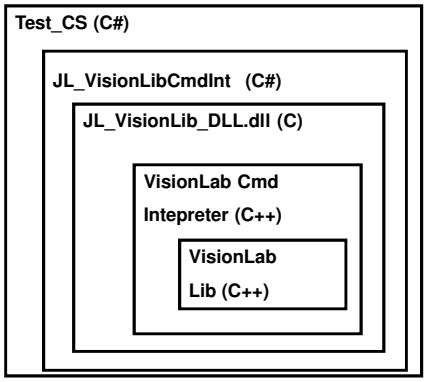
- **JL_VisionLib_DLL.dll:**
  wrapper around VisionLab command interpreter
- **JL_VisionLibCmdInt.cs:**
  .Net wrapper around JL_VisionLib_DLL
- **Test_CS:**
  example program in C#

Notes:

- all files can be found in the visual studio solution examples.sln
- In order to find the license file in the /windows/system32 directory:
set in the project property security:
- the "Enable ClickOnce Security Settings" to enabled
- set "This is a full trust application"

8/27/2018 VisionLab 178

Using VisionLab command interpreter in #C (*)



8/27/2018 VisionLab 179

JL_VisionLib_DLL.h (*)

```
/* This module is a dll wrapper around the VisionLib command interpreter.
 * It's function is to make the command interpreter accessable from non Visual
 * studio c++ languages.
 * The 4 main functions can be called using the windows standard calling conventions.
 * The result of executing a command or an error message from the Get/Set function
 * will returned in a buffer (answer) provided by the user. The user should allocated
 * enough memory. The size of this buffer is specified in parameter maxSizeAnswer.
 * If this buffer is not big enough the answer will be truncated.
 * For most functions a buffer size of 1024 bytes will be sufficient, but special commands like blobanalyse
 * can require more space. If the buffer is to small an error message will be stored in the answer.
 * JL_Execute and JL_SetImage will return the size of answer and/or error text.
 * JL_GetImage will return a handle to a bitmap or a 0 with an error message in answer if the
 * operation failed.
 * GetImage will produce a handle to a bitmap in Format32bppRgb.
 * SetImage will only accept a handle to a bitmap in Format32bppRgb.
 * GetBufPtr will return a pointer to the internal pixel array or a 0 with an error message in answer if the
 * operation failed. Note the buffer is still owned by the image and the buffer should not be deleted by
 * using the returned pointer.
 */

extern "C" __declspec(dllexport) int     JL_Execute   (const char *cmd, char * const answer,
                                        const int maxSizeAnswer);
extern "C" __declspec(dllexport) HBITMAP JL_GetImage  (const char *imageName, char* const answer,
                                        const int maxSizeAnswer);
extern "C" __declspec(dllexport) int     JL_SetImage  (const char *imageName, const char * JLImagetype,
                                        HBITMAP image, char* const answer, const int maxSizeAnswer);
extern "C" __declspec(dllexport) void*   JL_GetBufPtr (const char *imageName, char* const answer,
                                        const int maxSizeAnswer);
```

8/27/2018 VisionLab 180

JL_VisionLib_DLL.cpp (*)

You need to adapt the source if you want to use an another camera interface

```
/* How to add camera support to this component:
 * DXCam:
 * - add "RegisterDXCam(*cmdInt);" to DllMain
 * - add dxcam.lib from the ReleaseLib directory to this project
 * - build the project
 * Firewire FirePack:
 * - add "RegisterFirePackCam(*cmdInt);" to main
 * - add firepack.lib from the ReleaseLib directory to this project
 * - build the project
 * Gen<I>Cam with GenTL:
 * - add "RegisterJL_GenICam(*cmdInt);" to main
 * - add genicam.lib from the ReleaseLib directory to this project
 * - build the project
 */
```

8/27/2018 VisionLab 181

JL_VisionLib_DLL.cpp (*)

```
....

BOOL APIENTRY DllMain (HANDLE /*hModule*/, DWORD fdwReason, LPVOID /*lpReserved*/) {
  switch (fdwReason) {
    case DLL_PROCESS_ATTACH:
      cmdInt = new VisLibCmdInt(DefaultCallBackCmd(DefCallBackProc), maxHisSize, EchoOff);
      // ---------------------->    add here your extensions to the VisionLib command interperter
      //RegisterDXCam(*cmdInt);
      RegisterJL_GenICam(*cmdInt);
      //RegisterFirePackCam(*cmdInt);
      InsertUserOperCmds(*cmdInt);
      break;
    case DLL_THREAD_ATTACH:
      break;
    case DLL_THREAD_DETACH:
      break;
    case DLL_PROCESS_DETACH:
      if (cmdInt != 0) delete cmdInt;
      break;
  }
  return TRUE;  // Successful DLL_PROCESS_ATTACH.
} // DllMain

.....
```

8/27/2018 VisionLab 182

JL_VisionLibCmdInt.cs (*)

C# interface to VisionLab command intepreter

```
public class CmdInt  {
    protected const string dllName =
                        @"..\..\..\JL_VisionLib_DLL\release\JL_VisionLib_DLL.dll";
    public static string Init()
    public static string Execute(string cmd)
    public static Bitmap GetImage(string imageName)
    public static void SetImage(string imageName, string imageType, Bitmap bmp)
 }
```

Notes:

- dllName must point to the correct location of JL_VisionLib_DLL.dll
- Init must be call this as soon as possible to avoid late loading of dll
  If you change the current working directory before calling Init it is possible that the license file can not be found

8/27/2018 VisionLab 183

TestCS.prj (*)

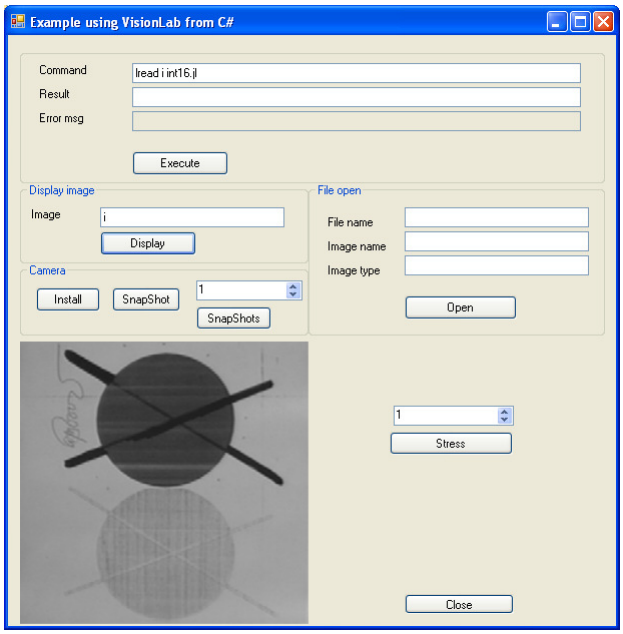
Test application in C#

Functions:

- **Test of one command**
- **Display VisionLab image in C# PictureBox**
- **Open an image in C# (*.bmp, jpg, etc), convert it to VisionLab format**
- **Install camera and make snapshots**
  Note: you will have to customize the action under the Install button according to the camera used
- **Stress test for testing on memory leaks**
  Note: C# is using a garbage collector. But if you do not free memory yourself, the garbage collector will select a moment to do it itself and may freeze your application for some time

8/27/2018 VisionLab 184

Screen shot of TestCS application (*)



8/27/2018 185