Computer Vision

Image math and geometric operators

10 April 2018

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

Image math and geometric operators

- **Image math**
- **Geometric operators**
  - **Rotate and translate**
  - **Mirror**
  - **Zoom**
  - **Warping, morphing and tweening**
  - **Find Corners Rectangle (Square)**
  - **Approximate Polygon (*)**
  - **SwapAxis and MapAxis (*)**
- **Miscellaneous**
  - **Copy**
  - **Convert**
  - **Insert (*)**
  - **ROI**
  - **ROIR (*)**
  - **SumColumns, SumRows**
  - **TransitionsColumns and TransitionsRows (*)**

Image math

- **Add (subtract) constant value to all pixels**
  - **Adjust brightness**
- **Add images**
  - **Extends exposure time (no Schwarzschild effect)**
  - **Average out distortions and noise**
- **Subtract images**
  - **Background elimination (logarithmic sensor)**
  - **Motion detection**
- **Multiply (divide) image with constant value**
  - **Adjust brightness**
- **Multiply images**
  - **Selection with use of mask image**
- **Divide images**
  - **Background elimination (linear sensor)**

27-8-2018 Image math and geometric operators 3

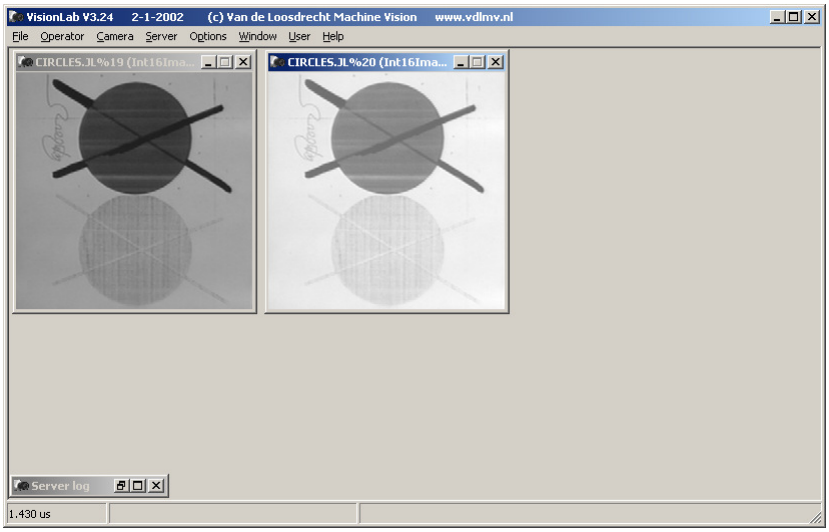
Demonstration Image math

- **Open image circles.jl**
- **Demo add pixel 80, use LUT = clip**
- **Close circles.jl**

- **Demo subtract images: motion detection of people in the audience note: camera is necessary !**

27-8-2018 Image math and geometric operators 4

Add Pixel 80



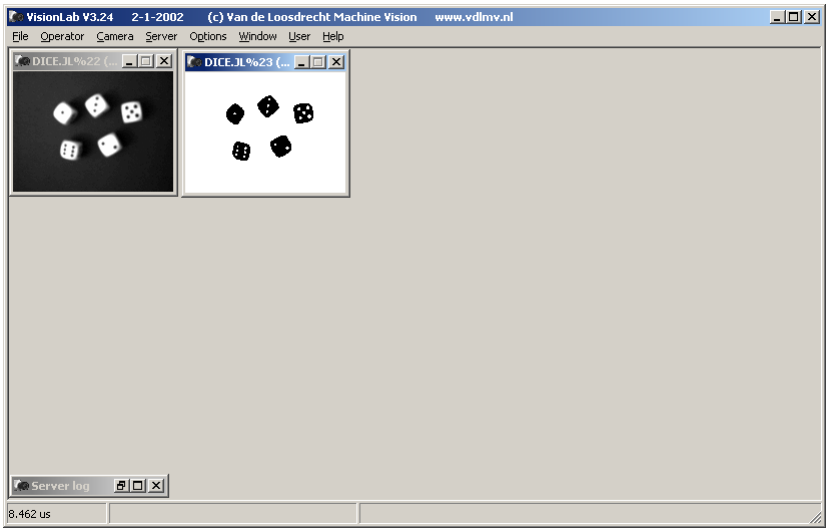
27-8-2018 Image math and geometric operators 5

Demonstration Image math usage of mask

- **Demo selection with use of mask image, get grey values of 'dice 6'**
  - **open image dice.jl**
  - **threshold 180 255**
  - **labels blobs, analyse pixels -> the six has labelnr 5**
  - **threshold 5 5**
  - **Fillholes (from segmentation menu)**
  - **multiply with original image:**
    - **everything is zero**
    - **the six has its original values**
- ***Do not close images, they are needed for a next example***

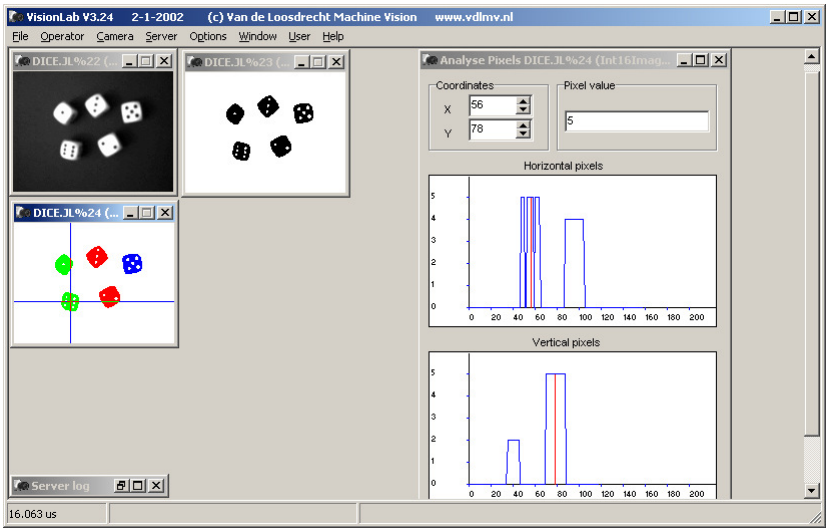
27-8-2018 Image math and geometric operators 6

Threshold 180 255



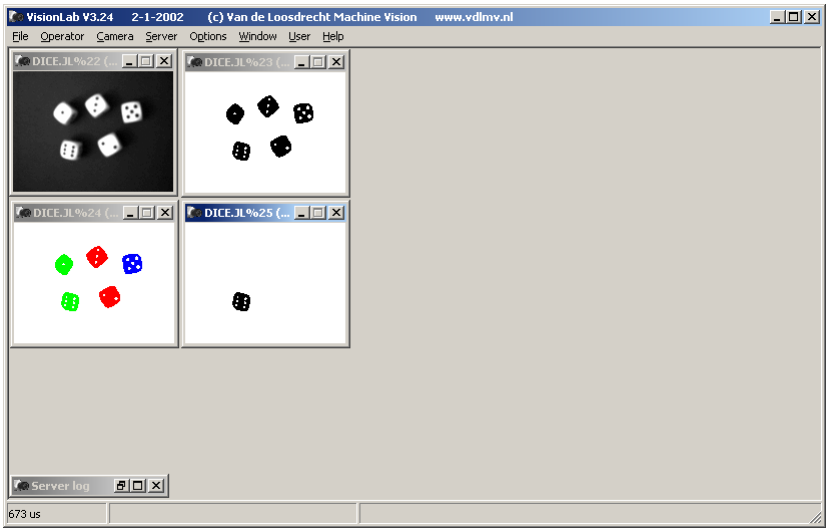
27-8-2018 Image math and geometric operators 7

Labels blobs, analyse pixels -> the six has labelnr 5

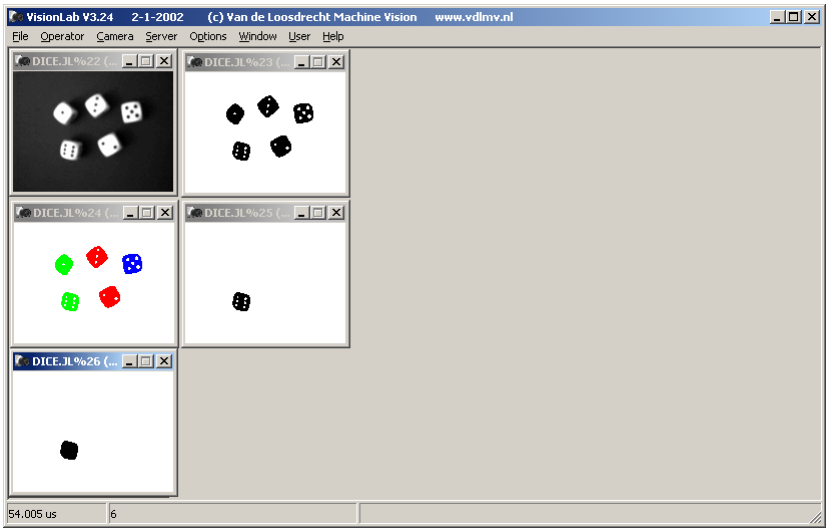


27-8-2018 Image math and geometric operators 8

Threshold 5 5



Fill holes



Multiply with original image

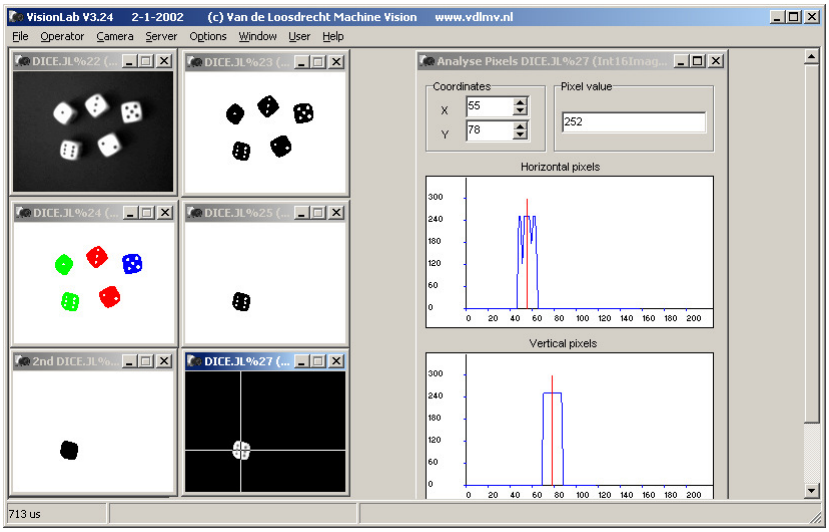


Image math for binary images

Binary images:

- **Background = 0**
- **Object = 1**

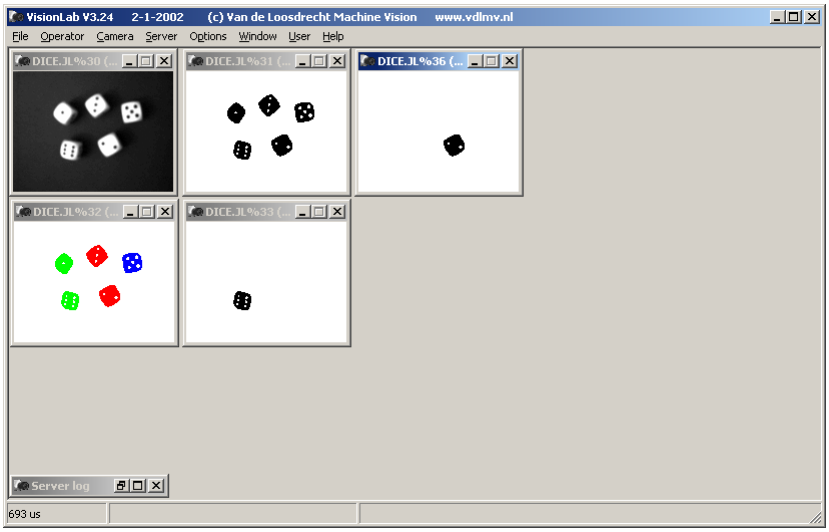
Used for masking operations:

- **And**
- **Or**
- **Exor**
- **Not**

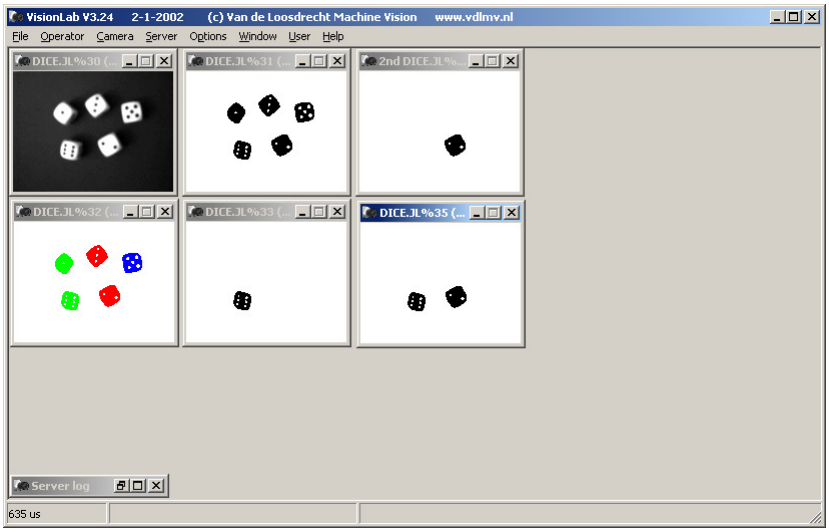
Demonstration Or operator

- **Demo or:**
  - **threshold 4 4 on labelled image, in order to select dice 'two'**
  - **or this image with binary image of the six**
  - **Note: adding has the same result if blobs do not overlap**

Threshold 4 4 on labelled image



'Or' this image with binary image of the six



27-8-2018 Image math and geometric operators 15

Image math

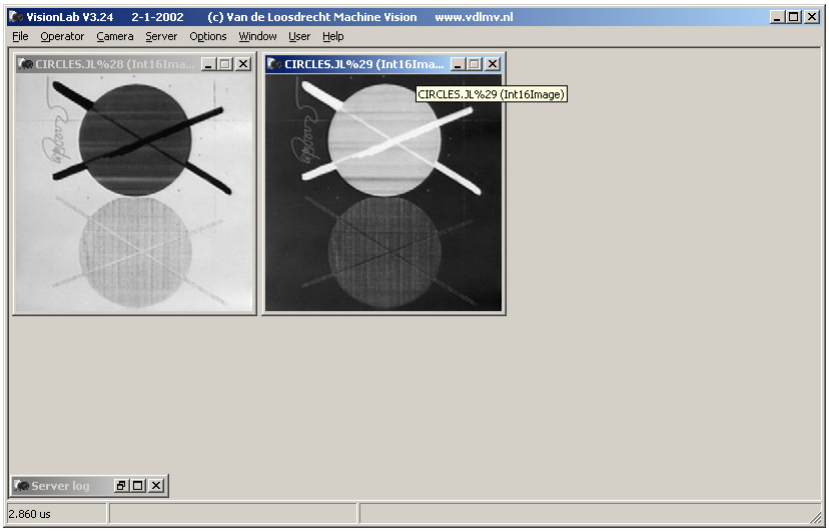
- **Abs (= absolute value for all pixels)**
- **Invert image**
- **Remainder images**
- **Min images**
- **Max images**
- **Mean images**
- **Modulo images**
- **ModuloPixel image pixel value**
- **Power imageX imageY**
- **PowerPixel image value**

27-8-2018 Image math and geometric operators 16

Demonstration Image math

- **Demo invert image on circles.jl**

Invert image



Exercise using masks

Use plate20.jl from *exercises directory* which has values in range [0..255]

a) write a script to change all pixels of the letters/digits/minus signs to zero, while leaving other pixels unchanged

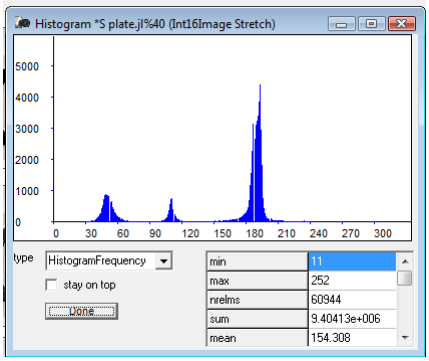
b) use the original image plate20.jl and write a script to add a pixel value of 80 to the background of the NL part, while leaving other pixels unchanged

c) take the result of assignment b, write a script to change pixel values of the plate to 128 while leaving other pixels (the letters/digits/minus signs) unchanged

Hint using masks

Look at histogram

- **Pixel values of the *letters* range from 0 to 70**
- **Pixel values of the *NL background* range from 70 to 130**
- **Pixel values of the *plate* values range from 130 to 255**



A) 

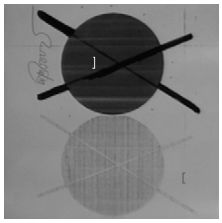
B) 

C) 

Answer: LicplateAnswer.jls

Exercise using masks (*)

Image h1.jl



Exercise using masks (*)

Use Int16Image h1.jl with has values in range [0..255]

- **a) write a script which changes all pixel with value 255 to 0, all other pixels are not changed**
- **b) write a script which changes all pixel with value 100 to 0, all other pixels are not changed**
- **c) write a script which changes all pixel with value 100 to 10, all other pixels are not changed**
- **d (*)) as c) write a script which replaces all pixels with a specified mask value by a specified new value and add this script as a new operator to VisionLab, the value of the maskpixel and the value of the new pixel value must be supplied as a parameter to the script**
- **e (*)) make your own c++ operator with functionality of d) and add it to VisionLab**

Exercise using masks (*)

see for answers:

- **h1a.jls**
- **h1b.jls**
- **h1c.jls**
- **h1d.jls + h1d.ini** **(note: script h1d.jls should be in current directory)**
- **e) to be done**

Background subtract versus division

Purpose: to correct an inhomogeneous illumination

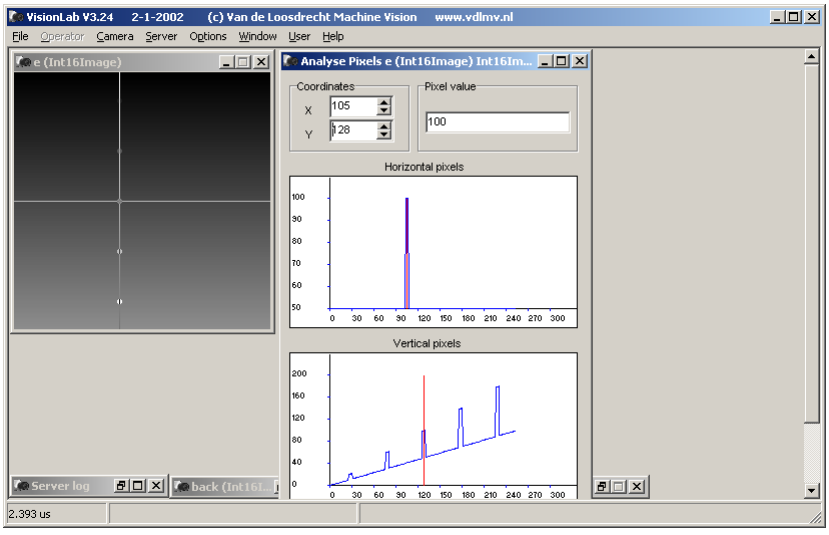
Strategies:

- **Logarithmic sensor: subtract images**
- **Linear sensor: divide images**

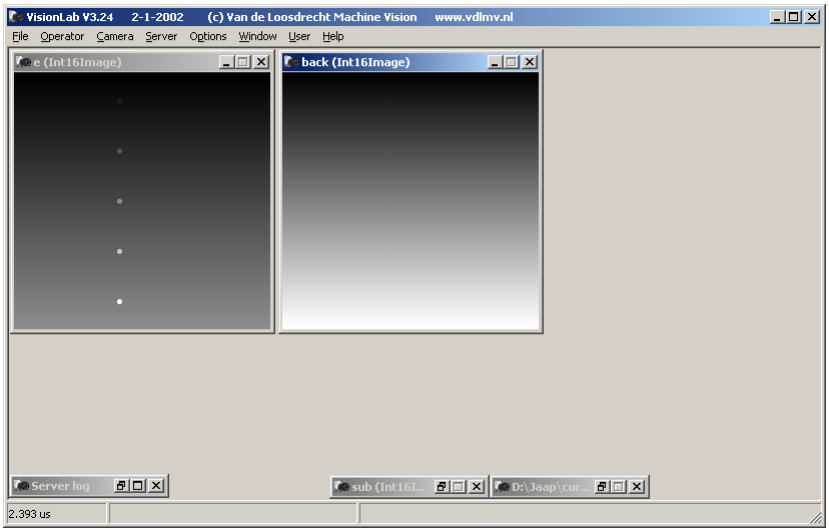
Demonstration background subtract versus division

- Note: MaximumFilter and MinimumFilter operations are explained later
- Open image backsubdiv.jl (or use script backsubdiv.jls)
- Demonstrate that thresholding is impossible, threshold 100 255
- subtract:
  - read e backsubdiv.jl
  - minimumfilter e em EdgeExtend octagon7x7
  - maximumfilter em back EdgeExtend octagon7x7
  - copy e sub
  - subtract sub back
  - see result with analyse pixels
  - Threshold 10 255 finds the dots
- divide:
  - convert e ef FloatImage
  - convert back backf FloatImage
  - // to avoid dividing zero and dividing by zero
  - addpixel ef 0.1 *// note: use . and not ,*
  - addpixel backf 0.1
  - copy ef divf
  - divide divf backf
  - see result with edit pixels

Analyse test image

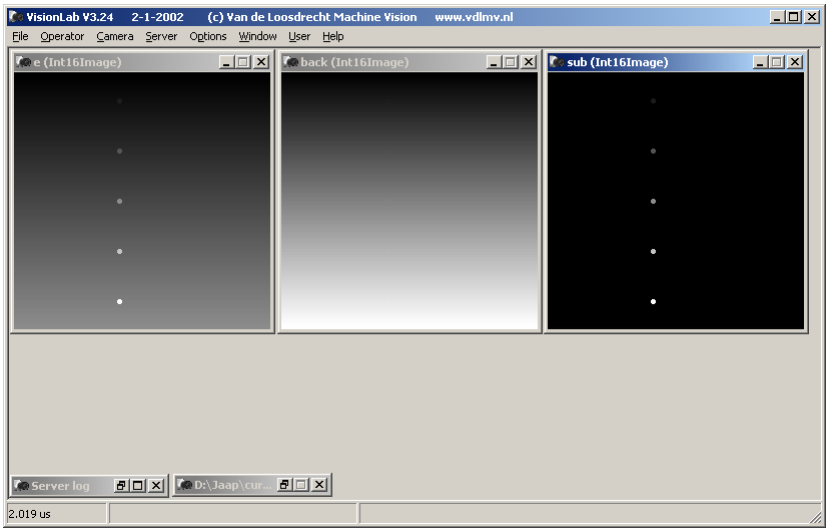


Generate background



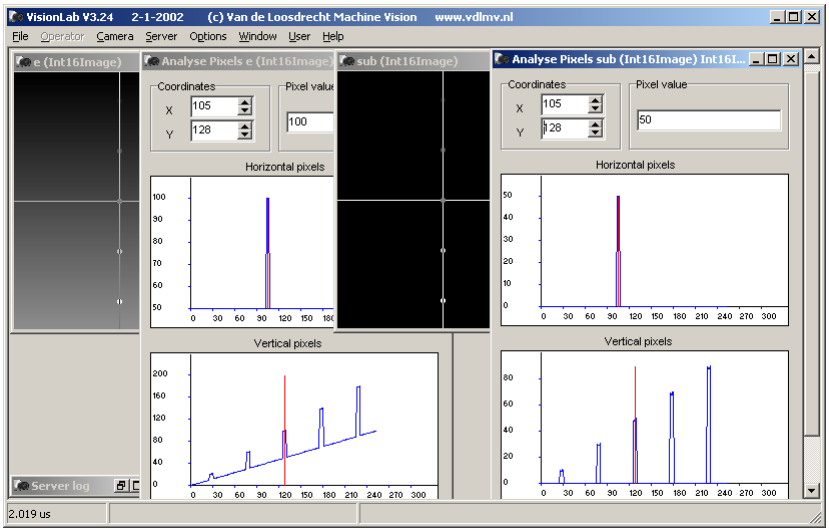
27-8-2018 Image math and geometric operators 27

Subtract background

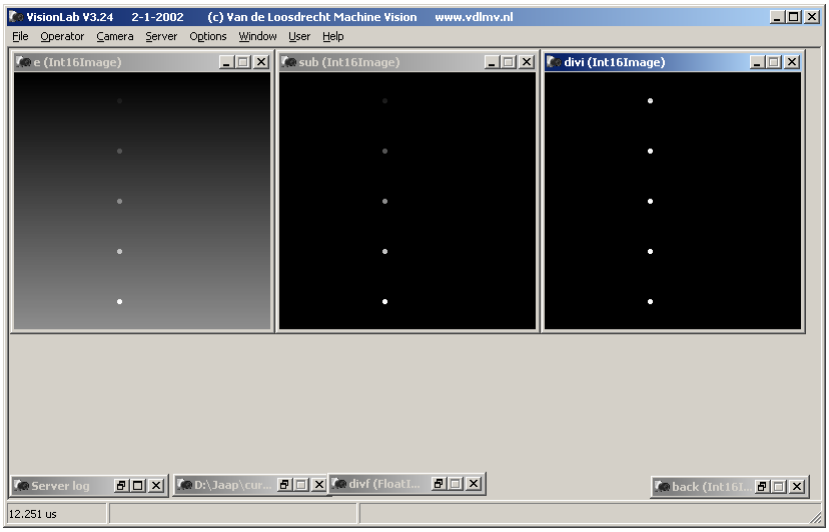


27-8-2018 Image math and geometric operators 28

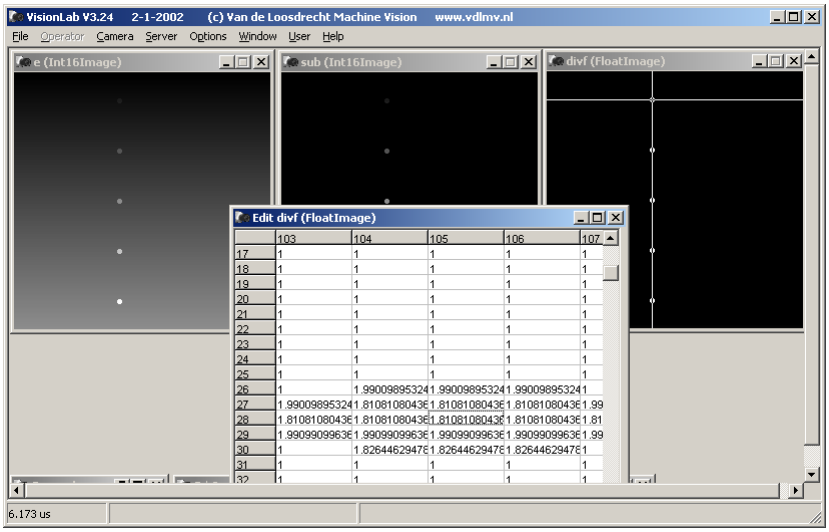
Analyse result subtraction



Divide with background

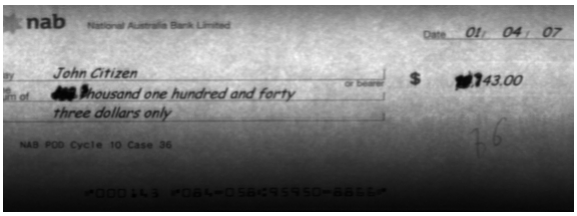


Analyse result division



Demonstration: background subtract versus division to do end the end of the chapter

Image cheque.jl taken with a line scan camera

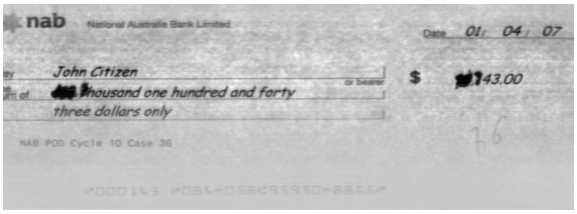


Correct the background using a subtract and a divide
This will be an exercise at the end of the chapter

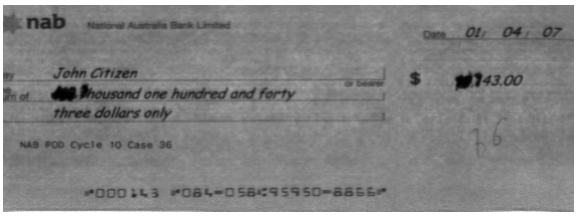
Answer: cheque_bg.jls

Solution: background subtract versus division

Subtract:
Cheque_subtract.jl



Divide:
Cheque_divide.jl



Answer: cheque_bg.jls

Geometric operations

Translation:

$x' = x + x_t$

$y' = y + y_t$

Rotation:

$x' = (x+x_c) * \cos(ß) - (y+y_c) * \sin(ß)$

$y' = (x+x_c) * \sin(ß) + (y+y_c) * \cos(ß)$

Problems

- **Image is rotated (translated) off the view area**
- **Pixel interpolation**
  - **Nearest pixel**
  - **Bilinear interpolation**

Geometric operations

Mirror:
- **In centre**
- **In X-axis**
- **In Y-axis**

Zoom (Scaling):
- **Zoom (height,width)**
- **ZoomXY (factorX,factorY)**
- **Reduce2 (Used for speeding up calculations)**
- **Enlarge2**
- **Binning (summing or averaging)**

Polar stretch:
- **This operator stretches and bends a part of a circle to a rectangular shape.**

Geometric operations

Warping:
- **Rubber band stretching**

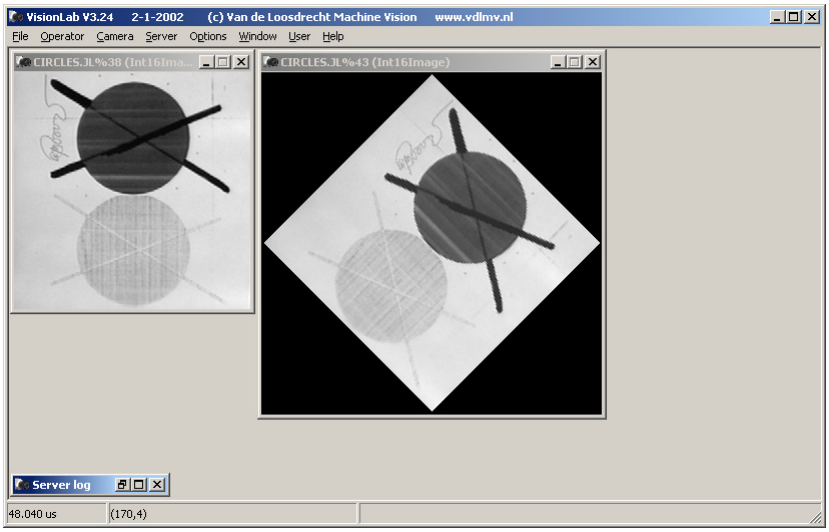
Morphing:
- **Repeatedly applying of warping**
- **Used mainly in computer graphics**

Tweening:
- **Calculate images between a start and stop image giving a smooth transformation**
- **Used mainly in video editing**

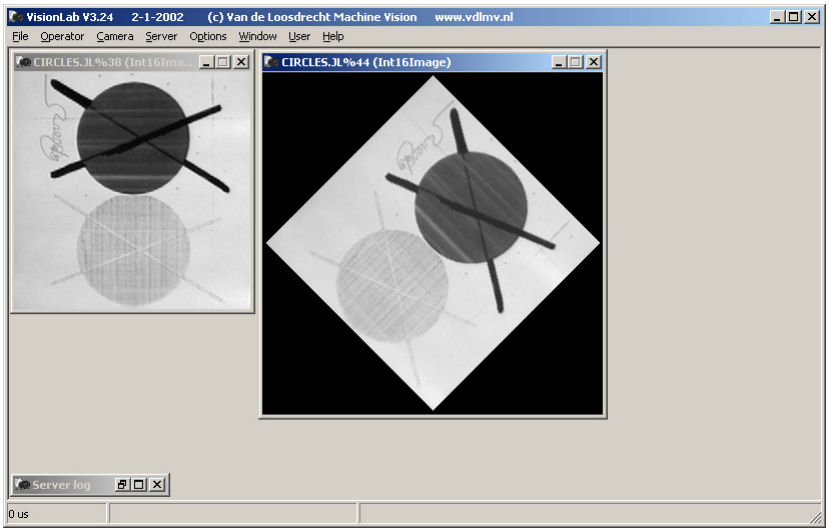
Demonstration geometric operations

- use image circles.jl
- demonstrate Rotate and RotateFull
- demonstrate difference between Nearest Pixel and Bilinear
- explain 'backwards' implementation
- demonstrate Mirror, Reduce2, Enlarge2, Zoom and ZoomXY
- Note: Enlarge2 is much faster then Zoom(2)
- Demonstrate Polar stretch on image polartest.jl with widget tool (or params 109 109 0 0 200 150 0 BilinearPixelInterpolation)
- Demonstrate Warp on image lic_plate1.jl with widget tool (or with warp_demo.jls)

Rotate Full with nearest pixel interpolation

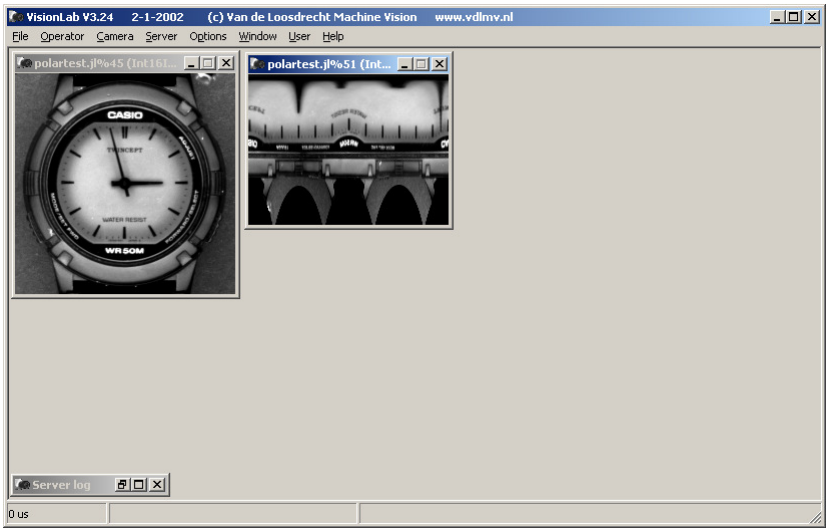


Rotate Full with bilinear pixel interpolation



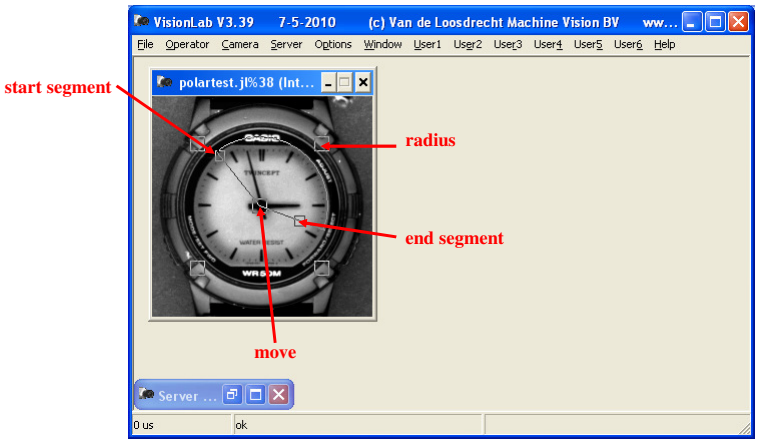
27-8-2018 Image math and geometric operators 39

Polar Stretch



27-8-2018 Image math and geometric operators 40

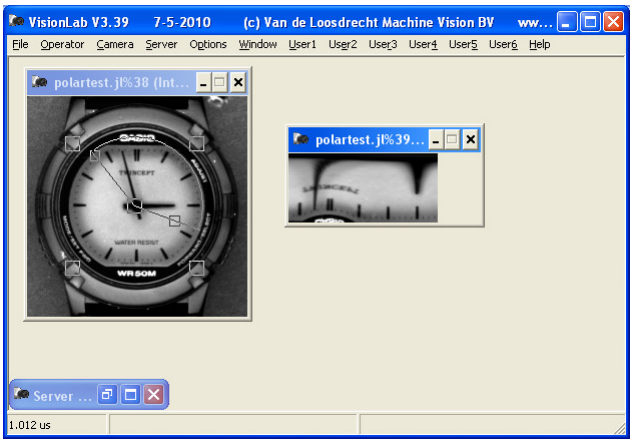
Polar Stretch using pie tool



Demo Polar Stretch using 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)**

Operator PolarStretch



Warp

Warp (srcImage, destImage, direction, leftTop, rightTop, leftBottom, rightBottom, height, width, border, interpolation)

Function result is false if 3 or more coordinates are (almost) in line

Direction: forward or reversed transform

Height and width are size of the destImage

Forward transform: perspective mapping of the quadranqle leftTop, rightTop, leftBottom and rightBottom in the srcImage to the total rectangle area of the destImage

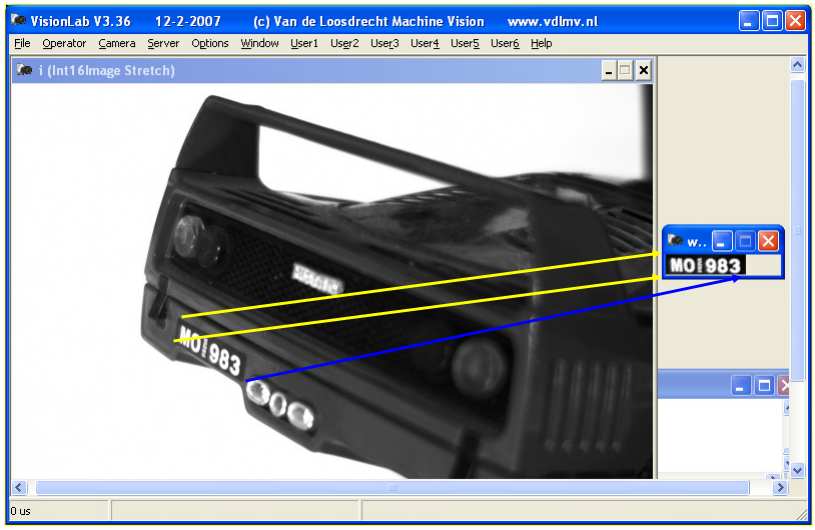
Inverse transform: inverse perspective mapping of the total rectangle area of the scrImage to the quadranqle in the destImage

Specified border value will be used as result pixel if information outside the source image is necessary for the calculation

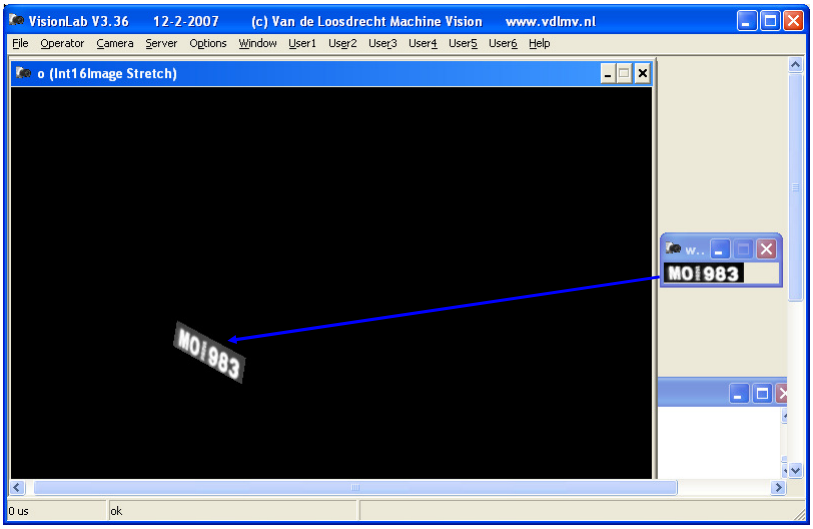
Demo Warp using grid tool

- **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**

Warp forward

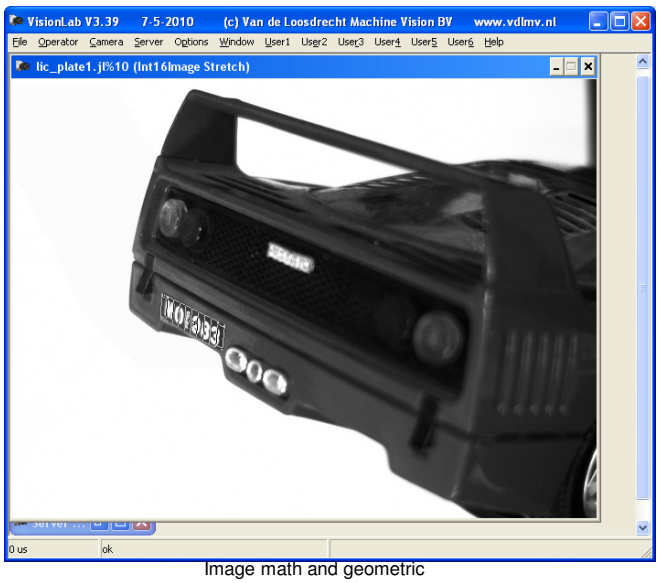


Warp Reverse



Warp using grid tool

Grid tool: drag landmarks to corners



Warp using grid tool: correct for warp

FindCornersRectangle

FindCornersRectangle (image, Connected, deltaPhi, Orientation, &$tab)

The FindCornersRectangle operator searches for for the biggest blob the corner points of a rectangle. This operator will allow a certain degree of deformation of the rectangle, like caused by perspective distortion. DeltaPhi specifies the degree of deformation allowed, deltaPhi is in radias and the max value is PI/4. This operator will work with rounded corners like usually seen in license plates.
Note: this operator will work only correctly if the shortest long side of the rectangle is longer then the two shortest sides. So it will NOT work for squares.

Connected: EightConnected or FourConnected
Orientation: Portrait or Landscape

The function result is true if rectangle is found. The array $tab will hold the coordinates in the order leftTop, rightTop, leftBottom and rightBottom

FindCornersRectangleSq (*)

FindCornersRectangleSq (image, Connected, &$tab)

The FindCornersRectangleSq operator searches for the biggest blob the corner points of a rectangle or a square.
This operator will allow a certain degree of deformation of the rectangle, like caused by perspective distortion. This algorithm is sensitive for distortions at the corner points and will not perform optimal with rounded corners.

Connected: EightConnected or FourConnected

The function result is true if rectangle is found. The array $tab will hold the coordinates in the order leftTop, rightTop, leftBottom and rightBottom

Demonstration FindCornersRectangle

- Open image 02LNGN.jpg
- Use ThresholdTool to select license plate (segment carefully!)
- FindCornersRectangle EightConnected 0.5 Landscape &$tab
- Warp ForwardT $tab[0] $tab[1] $tab[2] $tab[3] 50 300 0 NearestPixelInterpolation

- Same for FindCornersRectangleSq (*)

- Alternative:
- Convert image to HSV888Image
- ThresholdHSVchannels Int16Image 21 34 136 255 94 255
  (note: will be explained in chapter about color image processing)
- FindCornersRectangle EightConnected 0.5 Landscape &$tab
- Warp ForwardT $tab[0] $tab[1] $tab[2] $tab[3] 50 300 0 NearestPixelInterpolation

Demonstration FindCornersRectangle

RefineCornersRectangle

RefineCornersRectangle (image, &$tab, margin, &$newTab)

The RefineCornersRectangle operator finds the corner coordinates in subpixel precision of the quadrilateral described by the coordinates given in pixel precision. It returns the number of pixels found on the quadrilateral and the corner coordinates. Usually, it is used after FindCornersRectangle(SQ), and tries to improve its result.

It works for all types of quadrilaterals assuming they are perfect (straight lines), but it allows some curvature. It is robust and always gives an output, even if there is no quadrilateral in the image.

The arrays $tab and $newTab hold the coordinates in the order leftTop, rightTop, leftBottom and rightBottom.

margin: Half of the box width to be used for the FindEdgeLine operator.

RefineCornersRectangle explanation

This operator has as input a grayscale image and four points that approximately correspond to the corners of a rectangle in that image. For every two points that form an edge it places a box around the line which connects the two points. In this box, it uses the operator FindEdgeLine, which gives the line of the edge in subpixel precision. Then, it takes the intersections of those lines as the new corners.

If the initial corners are so close to the border that the box cannot be placed, the image is extended margin + 1 pixels on each side using the pixel in the border, before running the algorithm explained above. It returns the number of pixels found in the quadrilateral (results of all four FindEdgeLine added together).

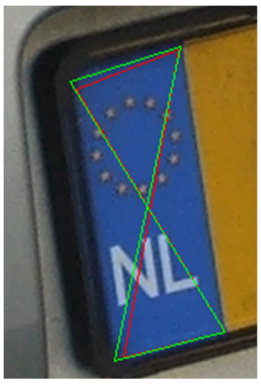
Demonstration RefineCornersRectangle

- **Open image testRefineCorners.jl**
- **ThresholdRGBChannels Int16Image 0 68 30 117 80 255**
- **FindCornersRectangleSQ EightConnected &$tab**
- **DrawPolygon image &$tab (255,0,0) KeepOriginal**

- **Extract1RGBchannel Blue Int16Image**
- **RefineCornersRectangle &$tab 10 &$newTab**
- **DrawPolygon image &$newTab (0,255,0) KeepOriginal**

See script refineCornersRectangle.jls

Demonstration RefineCornersRectangle



RefineCornersRectangle
FindCornersRectangleSQ

Approximate Polygon (*)

**ApproxPolygon (image, connected, minDistance,
maxVertices , &$tab)**

The ApproxPolygon operator calculates the approximate polygon of the biggest blob.

Connected: EightConnected or FourConnected

MinDistance: the approximation search will stop if all pixels on the contour are closer minDistance pixels to the polygon, ,minimum value is 1

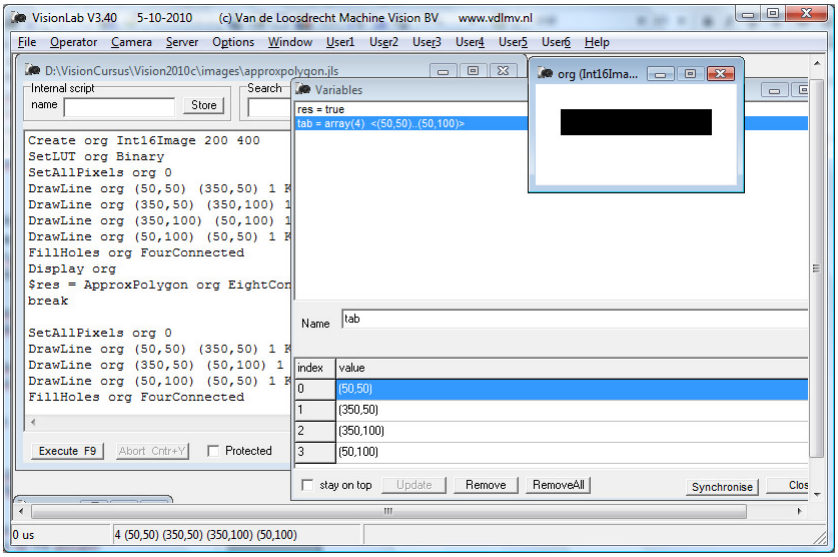
MaxVertices : the number of vertices for the polygon.
If maxVertices > 0 search for maximal max number of vertices
If maxVertices == 0 then keep searching until maxD < minDistance

The function result is the number of vertices found.
The array $tab will hold an array with coordinates for the polygon.
The coordinates are sorted in clock wise orientation.

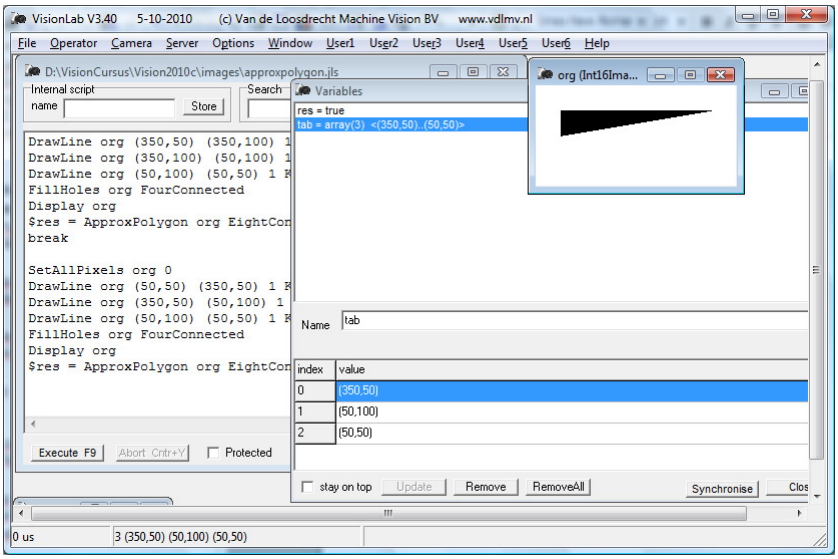
Demonstration ApproxPolygon (*)

- **Run script approxpolygon.jls**
  - **Find corners rectangel**
  - **Find corners triangle**
- **NOTE: The coordinates are sorted in clock wise orientation**

ApproxPolygon to find corners rectangle (*)



ApproxPolygon to find corners triangle (*)

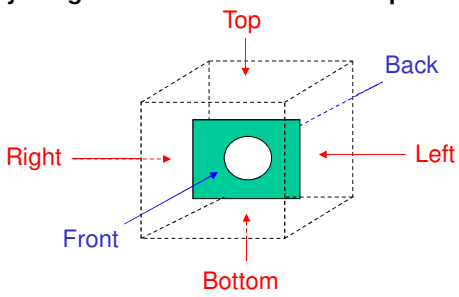


SwapAxis (*)

Representation of a 3D binary image (Source 3D camera) in a 2D image

- **X axis of the image = X axis of the 3D image**
- **Y axis of the image = Y axis of the 3D image**
- **Pixel value of the image = Z axis of the 3D image**

A 3D binary image can be viewed from multiple viewpoints.



27-8-2018 Image math and geometric operators 63

Time of flight camera (*)

Time of flight camera produces 3D binary images

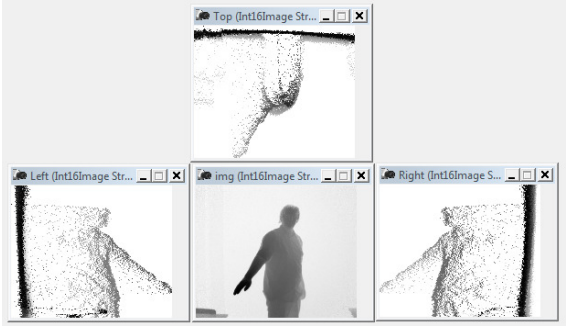
27-8-2018 Image math and geometric operators 64

SwapAxis (*)

SwapAxis (srcImage, dstImage, viewPoint, scale)

Generates a new image by viewing the 3D binary image from a different viewpoint (Left, Right, Back, Top, Bottom, Front)

Scale can be used to reduce the destination image dimensions. The Z axis (= pixel value) is divided by the scale.



27-8-2018 Image math and geometric operators 65

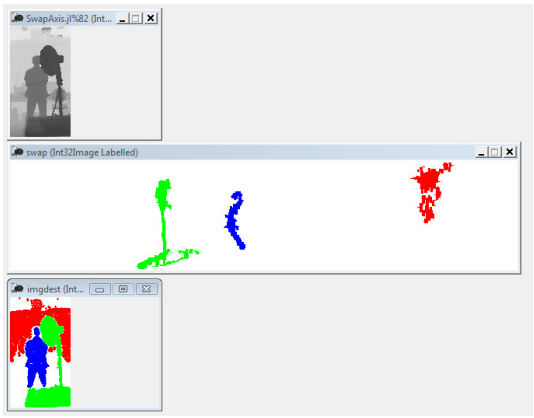
Demonstration SwapAxis (*)

- **Run script persoon3d.jls**

27-8-2018 Image math and geometric operators 66

SwapAxis Example (*)

Perform computer vision filters on the y/z or x/z plane of an image to find objects in 3D.



Script: swapaxis.jls

MapAxis (*)

MapAxis (srcImage, dstImage, axisMapping)

The MapAxis operator initializes a destination image by mapping the x-axis and y-axis of the source image to the positive or negative x-axis or y-axis of the destination image.

The axisMapping parameter specifies the predefined mapping performed. The operator can be used to rotate and/or mirror the source image in one pass. Possible values:

Rotate90	Rotate180	Rotate270
MirrorInXAxis	MirrorInYAxis	
Rotate90AndMirrorInYAxis	Rotate90AndMirrorInXAxis	
Copy	MirrorInCenter	

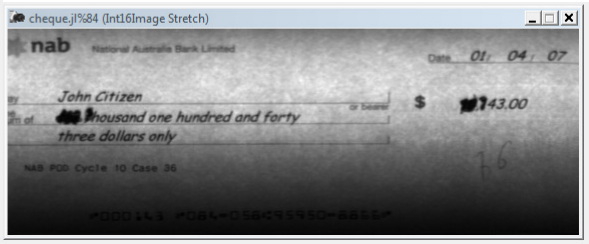
MapAxis Example (*)

Equation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} xx & xy \\ yx & yy \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} ww & wh \\ hw & hh \end{pmatrix} \times \begin{pmatrix} w-1 \\ h-1 \end{pmatrix}$$

axisMapping = Rotate90

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} w-1 \\ h-1 \end{pmatrix}$$



Miscellaneous

Copy:

- **Make a copy of image**

Convert

- **Convert image to another type:**
  ByteImage, ComplexFloatImage, ComplexDoubleImage, DoubleImage, FloatImage, HSV888Image, HSV161616Image, Int8Image, Int16Image, Int32Image, RGB888Image, RGB161616Image, YUV888Image andYUV161616Image.

Miscellaneous

ROI:
- **Region Of Interest**
- **copy aligned rectangle from image**

ROIR:
- **Region Of Interest Rotated**
- **copy arbitrary rotated rectangle from image**

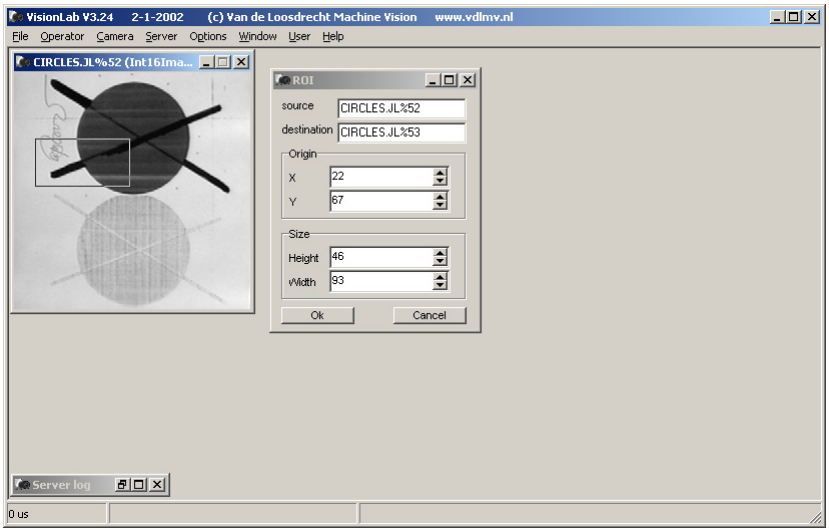
SumColumns and SumRows:
- **The columns or rows of the image are summed**

Insert:
- **Insert image in another image**

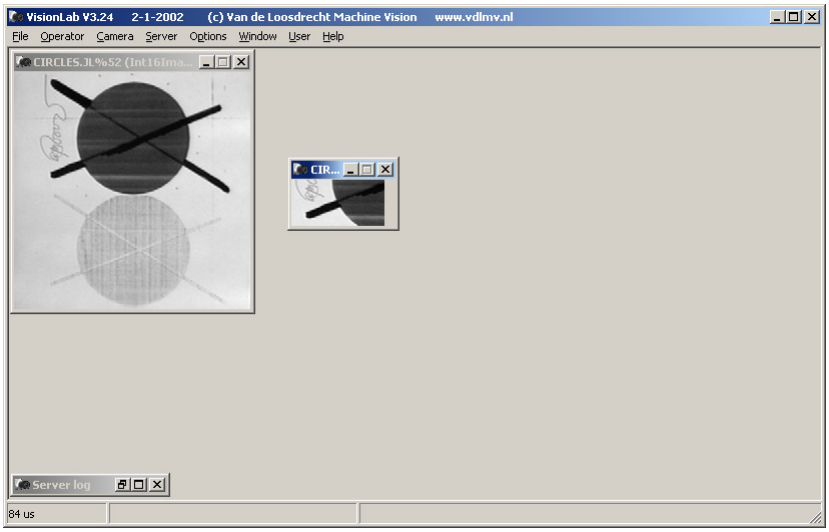
Demonstration Convert and ROI

- **Use image circles.jl**
- **Demonstrate Convert (no slides)**
- **Demonstrate ROI**

Demonstration of ROI



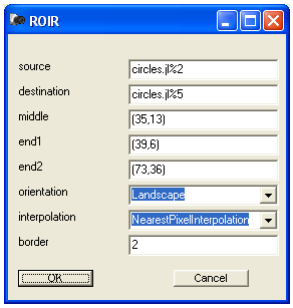
Demonstration of ROI



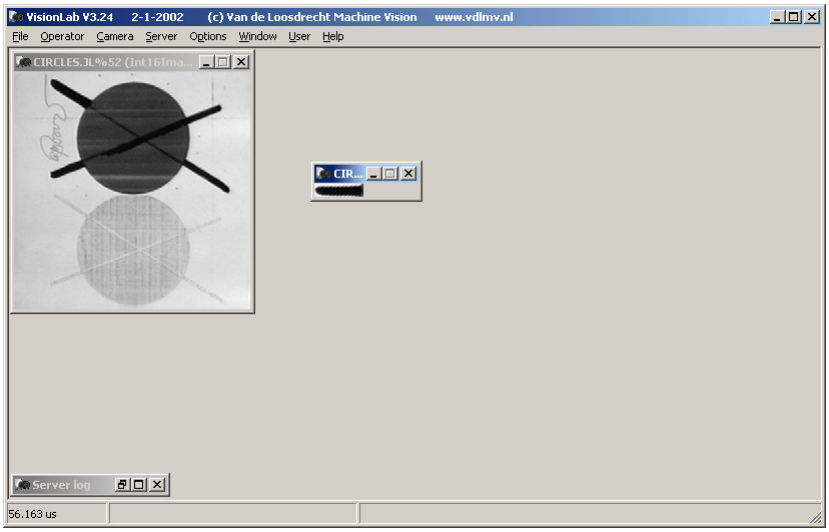
Demonstration ROIR and Insert (*)

- **Use image circles.jl**
- **Demonstrate ROIR**
  - **ROIR (35,13) (39,6) (73,36) Landscape BilinearPixelInterpolation 2**
- **Demonstrate Insert, insert result of ROIR somewhere in circles**

ROIR parameters (*)

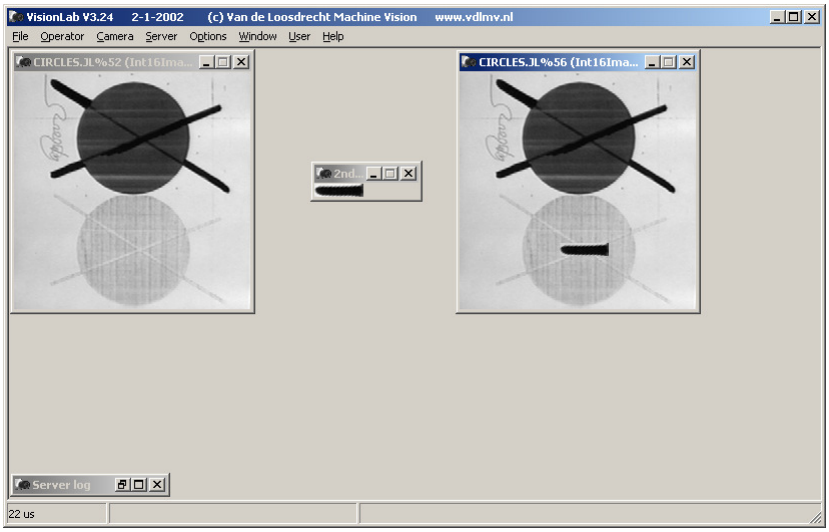


ROIR (*)



27-8-2018 Image math and geometric operators 77

Insert (*)



27-8-2018 Image math and geometric operators 78

SumColumns and SumRows

SumColumns and SumRows:

- **The columns or rows of the image are summed into an image with height or witdh of 1 pixel**

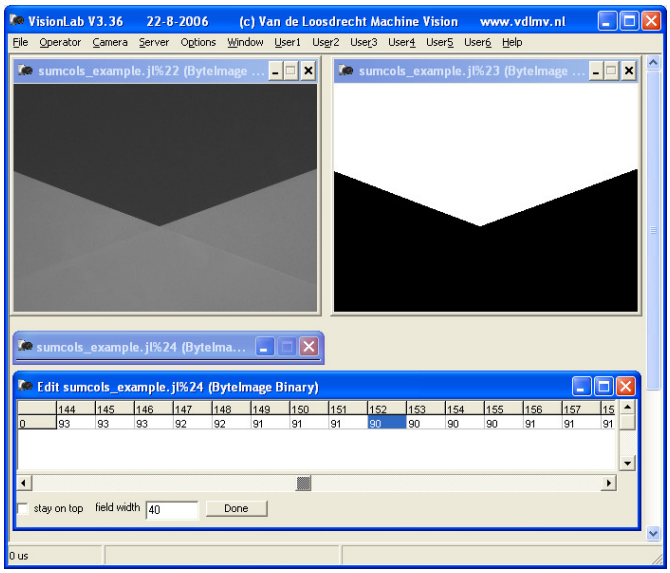
Usage:

- **For creating "thickness profiles"**

Demonstration SumColumns

- **Open sumcols_example.jl**
- **ThresholdIsoData BrightObject**
- **SumColumns (from Point menu)**
- **Set LUT to Stretch**
- **Show result of SumColumns with Edit**

Demonstration SumColumns



TransitionsColumns and TransitionsRows

TransitionsColumns src destination threshold
TransitionsRows src destination threshold

The columns or rows of the src image are scanned for transitions. The number of transitions are stored in the destination image.
A transition is defined as:

- **a pixel has value lower then threshold and its neighbour column or row pixel has a value greater or equal threshold**

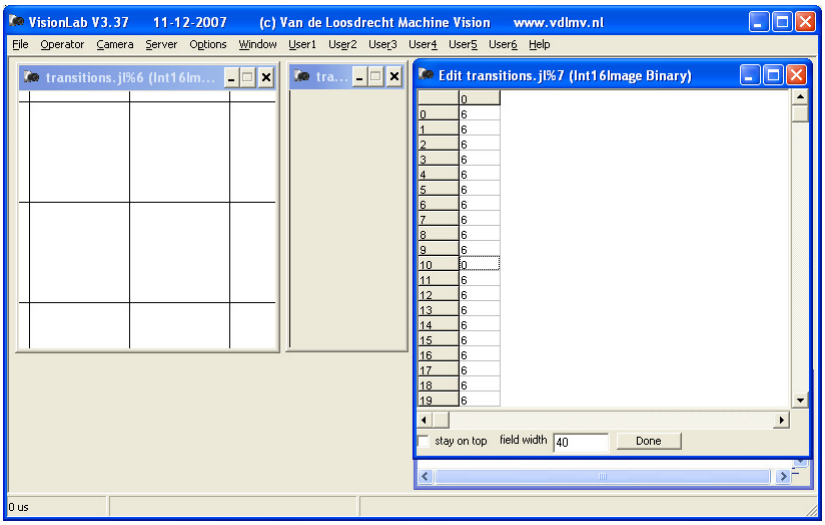
or

- **a pixel has value greater or equal then threshold and its neighbour column or row pixel has a value lower threshold**

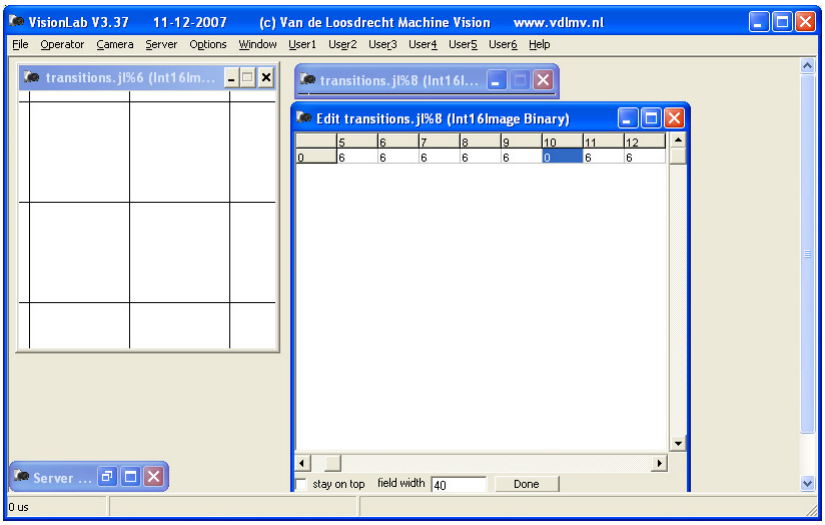
Demonstration TransitionsColumns/Rows (*)

- Open transitions.jl
- TransitionsRows 1
- Examine result with Edit image
- TransitionsColumns 1
- Examine result with Edit image

TransitionsRows (*)

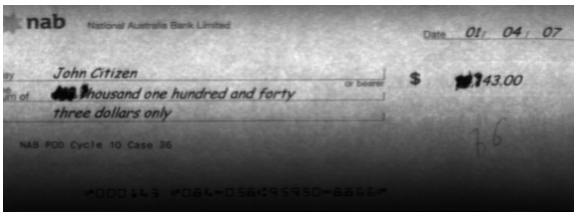


TransitionsColumns (*)



Exercise: background subtract versus division

Image cheque.jl taken with a line scan camera

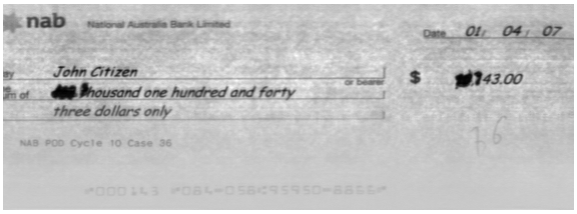


Correct the background using a subtract and a divide
Hint: Use ROI and ExtendBorder to generate a background image

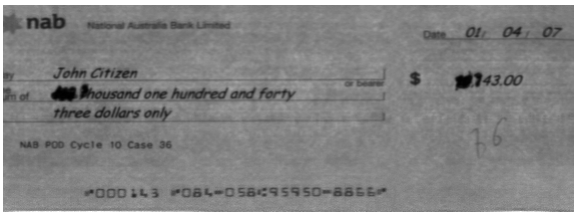
Answer: cheque_bg.jls

Solution: background subtract versus division

Subtract:
Cheque_subtract.jl



Divide:
Cheque_divide.jl



Answer: cheque_bg.jls

Exercise (*)

- **Experiment with the basic operators**
- **The basic operators will be needed in other exercises**