NHL STENDEN
computer vision & data science

Computer Vision

Edge detection

10 April 2018

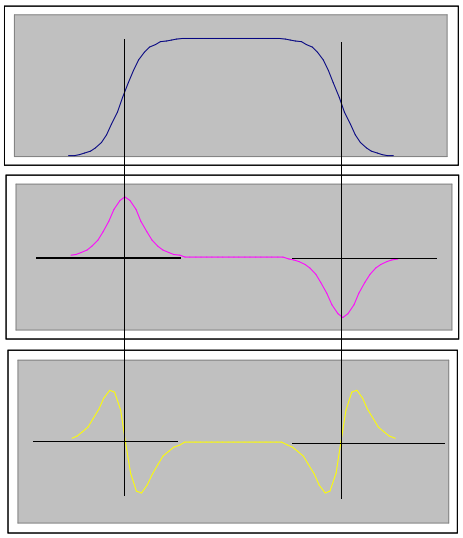
j.van.de.loosdrecht@nhl.nl, jaap@vdlmv.nl

Edge detection

Overview:

- First derivative
  - Gradient difference
  - Template matching (*)
- Second derivative
  - Laplacian
  - Laplacian of Gaussian (Mexican Hat)
  - Difference of Gaussians (*)
- Combination of first and second derivative
- Connecting edges
  - Marr- Hildreth
  - Canny (*)
- FindEdgeLine
- FindSubEdge(s)OnLine
- FindEdgeCircle
- FindFirstTransitions (*)
- FindFirstEdges (*)

First and second derivatives of an edge



Edge profile

First derivative

Second derivative

27-aug-18 Edge detection 3

Edge detection

First derivative :

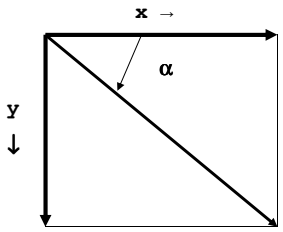
- **Gradient difference**
  - **Sobel**
  - **Prewitt**
  - **Frei Chen**
  - **Scharr**
  - **Roberts (*)**
- **Template matching (*)**
  - **Kirsch**
  - **Robinson**

27-aug-18 Edge detection 4

Edge detection

Definitions:

- **magnitude: strength of edge**
- **direction: orientation of edge**
- **angles are measured in radians (- π .. π]**



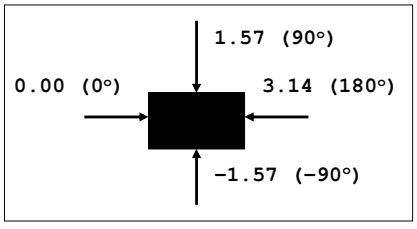
27-aug-18 Edge detection 5

Edge detection

Definitions (continued):

- **edge directions are calculated from low towards high pixel values perpendicular at the edge contour**
  example:

White = 0, Black = 10 !!!

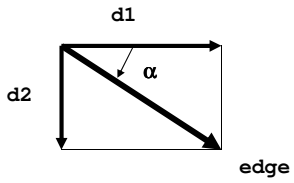


27-aug-18 Edge detection 6

Gradient difference

Idea:

- **Calculate the edge using convolution in two perpendicular directions d_1 and d_2**
- **Edge magnitude = sqrt ($d_1^2 + d_2^2$)**
- **Edge direction = arctan (d_1 / d_2)**



27-aug-18 Edge detection 7

Gradient difference

Examples:

- **Sobel (src, magImage, dirImage, gradient, dirScale, minedge);**

  gradient: magnitude, direction or both
  dirScale: direction in radians * dirScale
  minEdge: if gradient is both, all directions with an edge magnitude lower then minEdge are not calculated and set to zero.

  Masks:

```
-1  -2  -1          -1   0   1
 0   0   0          -2   0   2
 1   2   1          -1   0   1
```

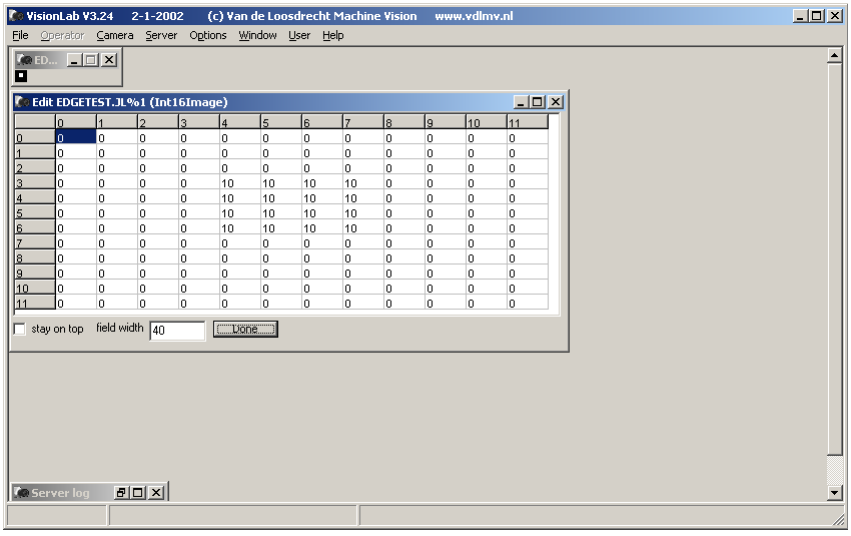
27-aug-18 Edge detection 8

Demonstration Sobel

- **Open image edgetest.jl** ***(do not use sq2.jl)***
- **Apply Sobel magnitude on image**
- **Apply Sobel direction with scale = 10000 on image**
- **Edit images (3x) and explain**

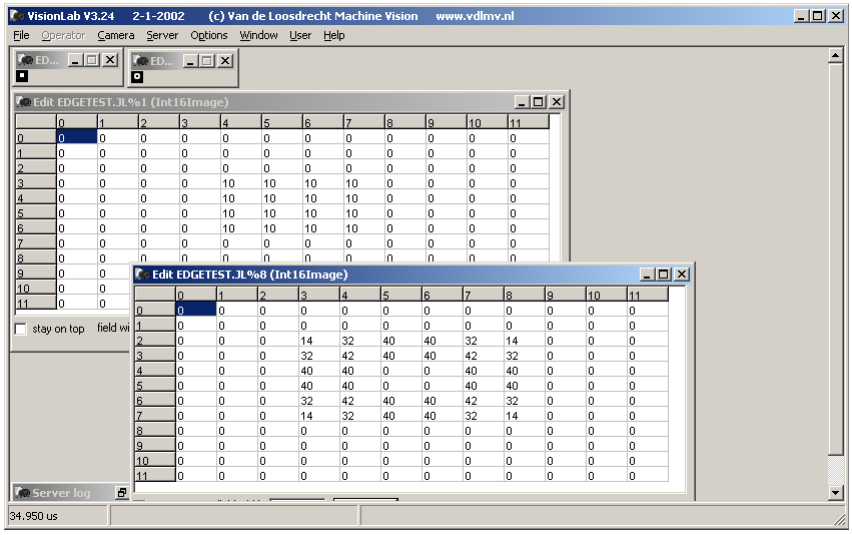
27-aug-18 Edge detection 9

Image edgetest.jl



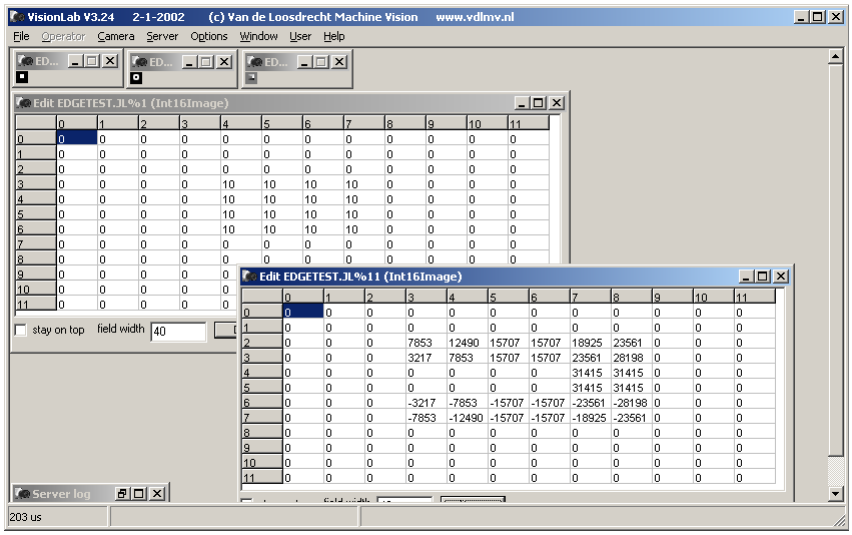
27-aug-18 Edge detection 10

Apply Sobel magnitude



27-aug-18 Edge detection 11

Sobel direction with scale = 10000



27-aug-18 Edge detection 12

Exercise Find strong edges with angle of 90°

Make a script for image circles.jl that finds strong edges (> 200) with angle of 90° (p.a.: bottom of dark circle) using Sobel edge detection

Answer: strongedges2.jls

Gradient difference

Examples (continued):

- **Prewitt:**

$$\begin{matrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix} \qquad \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$$

- **Frei Chenn:**

$$\begin{matrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{matrix} \qquad \begin{matrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{matrix}$$

implemented with ints and divisor of 100

- **Sharr:**

$$\begin{matrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{matrix} \qquad \begin{matrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{matrix}$$

theoretical best accuracy, but beware of overflow

Gradient difference (*)

Examples (continued):

- **Roberts:**

```
−1  0          0 −1
 0 −1          1  0
```

notes:

- **mask centre is top left.**
- **all edges are shifted by one-half of a pixel in x and y direction.**
- **the two diagonal directions are rotated by $\pi/4$.**
- **due to smaller masks faster operation**

27-aug-18 Edge detection 15

Template matching (*)

Idea:

- **Calculate N times the edge using convolution with mask that is rotated 2π / N after each convolution**
- **Edge magnitude = max ($conv_i$: i = 1 to N)**
- **Edge direction = rotation of max ($conv_i$: i = 1 to N)**

problem: what to do if two or more masks give the highest value

27-aug-18 Edge detection 16

Template matching (*)

Examples:

- **Kirsch: (8 rotations)**

```
−3 −3  5
−3  0  5
−3 −3  5
```

- **Robinson: (8 rotations)**

```
−1  0  1
−2  0  2
−1  0  1
```

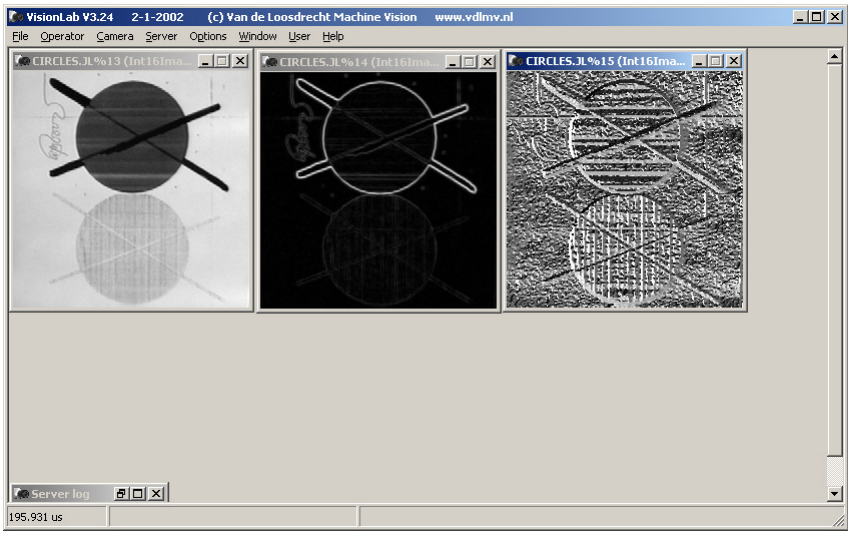
27-aug-18 Edge detection 17

Demonstration Kirsch (*)

- **Apply Kirsch on circles.jls for magnitude and direction, note increase of processor time compared with Sobel.**

27-aug-18 Edge detection 18

Kirsch on circles.jls for magnitude and direction (*)



27-aug-18 Edge detection 19

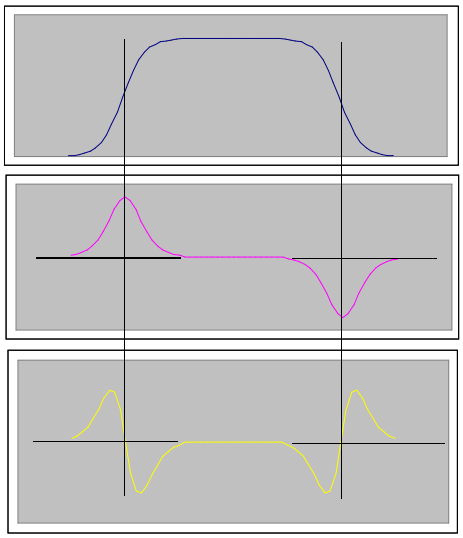
Edge detection

Second derivative

- **Laplacian**
- **Laplacian of Gaussian (Mexican Hat)**
- **Finding edges using zero crossings**

27-aug-18 Edge detection 20

First and second derivatives of an edge



Edge profile

First derivative

Second derivative

27-aug-18 Edge detection 21

Laplacian

Examples of convolution masks:

- **Laplacian 3x3:**

```
-1  -1  -1
-1   8  -1
-1  -1  -1
```

- **Laplacian 5x5:**

```
 0   0  -1   0   0
 0  -1  -2  -1   0
-1  -2  16  -2  -1
 0  -1  -2  -1   0
 0   0  -1   0   0
```

Usage:

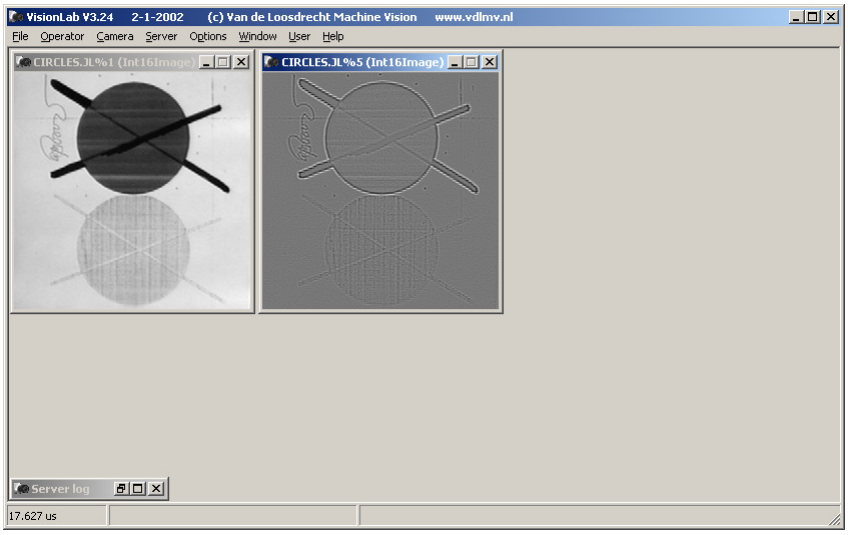
- **high pass filter**
- **edge detection, but sensitive to noise**

27-aug-18 Edge detection 22

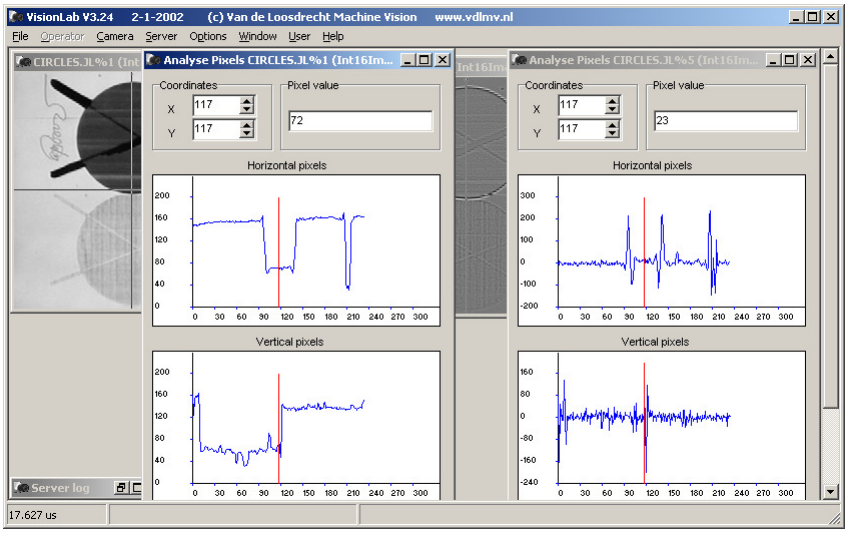
Demonstration high pass filter

- Open image circles.jl
- Convolution with Laplacian 3x3
- 2x analyse pixels:
  - low frequencies -> 0
  - high frequencies -> |pixel value| >> 1

Convolution with Laplacian 3x3



original **after convolution with Laplacian**



Mexican hat

- **Variant of low and high pass filters**
- **Combination of low and high pass filter**
- **Mask (7x7):**

```
 0   0  -1  -1  -1   0   0
 0  -1  -3  -3  -3  -1   0
-1  -3   0   7   0  -3  -1
-1  -3   7  24   7  -3  -1
-1  -3   0   7   0  -3  -1
 0  -1  -3  -3  -3  -1   0
 0   0  -1  -1  -1   0   0
```

- **Local noise is smoothed out by low pass filter in centre**

Laplacian of Gaussian

LoGFilter (image, sigma, size)

This is a generalized implementation of a Mexican hat filter.

**Parameter sigma is the standard deviation,
typical values are [2/3 .. 3].**

**Size is the size of the neighbourhood of the operation.
If size is 0 the algorithm calculates a size so that pixels at 3*sigma are neglected.**

Difference of Gaussians (DoG) filter (*)

DoGFilter (image, sigmaLow, sigmaHigh, size)

An alternative implementation for a generalised Mexican hat filter, using the difference of two Gaussians with substantially different sigmas.

Parameters:

- **sigmaLow and sigmaHigh are the standard deviations for the DoG operator. Typical values are [0 .. 3].**
- **size is the size of the neighbourhood of the operation. If size is 0 the algorithm calculates a size so that pixels at 3*sigma are neglected**

Demonstration Mexican hat (*)

- Open image circles.jl (use LUT stretch)
- Add noise 1 0 50
- Convolution Laplacian 5x5 on noise image
- Convolution Mexican hat on noise image
  (smooth noise and enhance high frequencies)

- Open image circles.jl
- Convolution Mexican hat on image
- LoGFilter 1.4 7 on Image (same result)

With noise **Mexican hat** **Laplacian (*)**

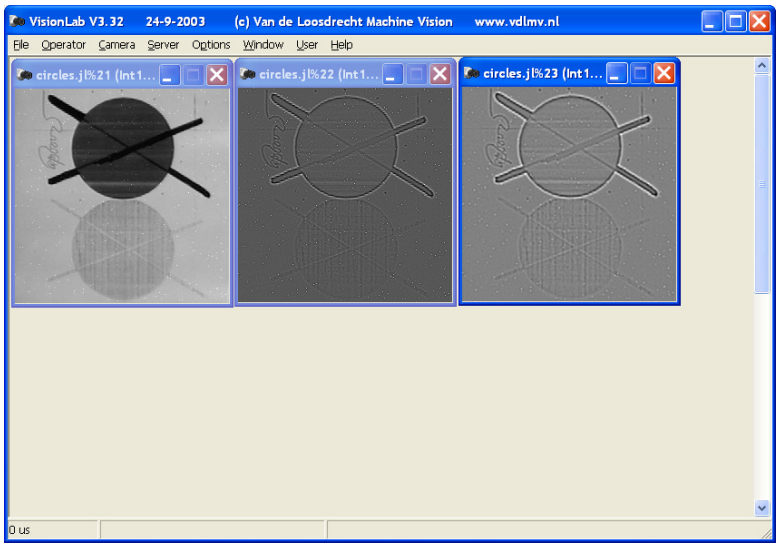
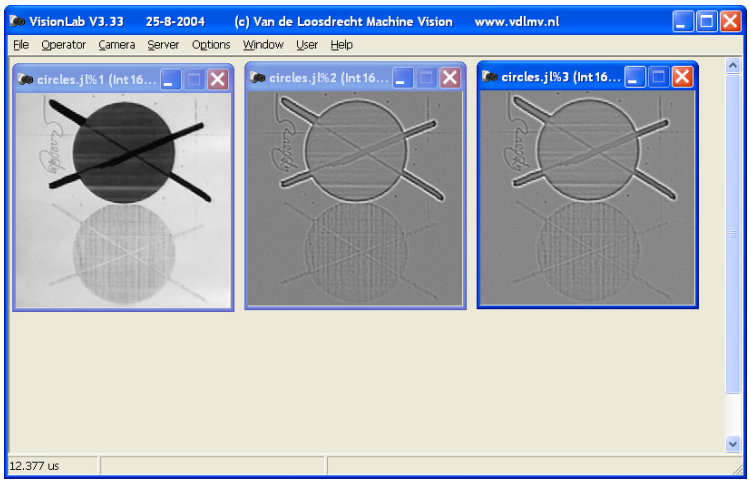


image MexicanHat7x7 LoGFilter 1.4 7 (*)



27-aug-18 Edge detection 31

Combination of first and second derivative

The positions of the edges can not be found exactly using first derivative edge detectors because the maximum value are position dependant.

The zero crossings in the second derivative 'ANDed' with the strong edges in the first derivative give the exact position of the edges.

27-aug-18 Edge detection 32

Zero crossings

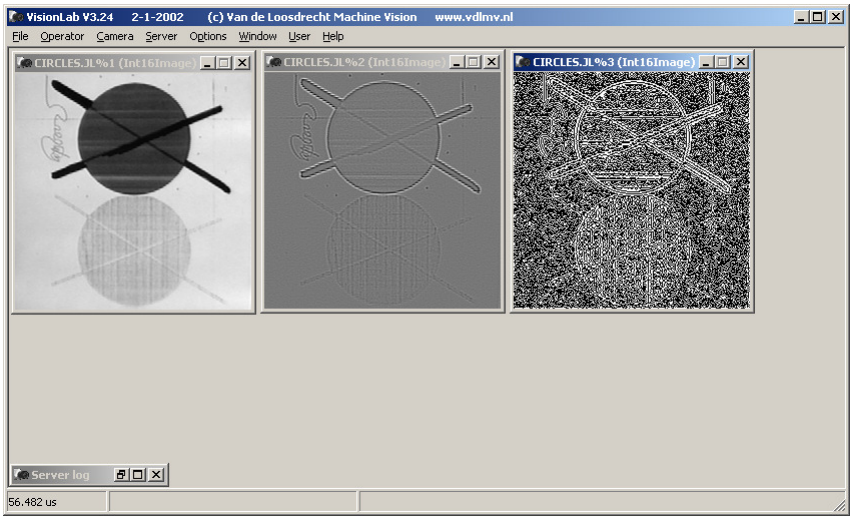
- Open image circles.jl
- Convolution Laplacian 3x3
  - Show zero crossings with pixel analysis
- Zero crossings FourConnected on laplacian
- Sobel Magnitude on circles
- Threshold 200 1000 on sobel
- Multiply zero crossing with threshold gives the exact position of the edges

Note:

- With zero crossings the "middle" of the edge will be found and the edge will be approximately one pixel thick
- With Sobel magnitude followed by threshold an edge will be found which is in general more then one pixel thick, its thickness will vary with the intensity of the lighting.

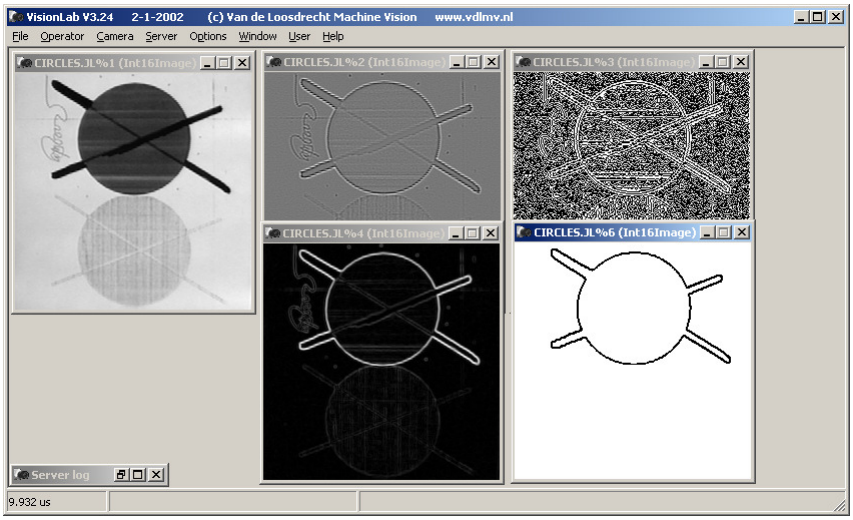
27-aug-18 Edge detection 33

original **Laplacian** **Zero crossings**



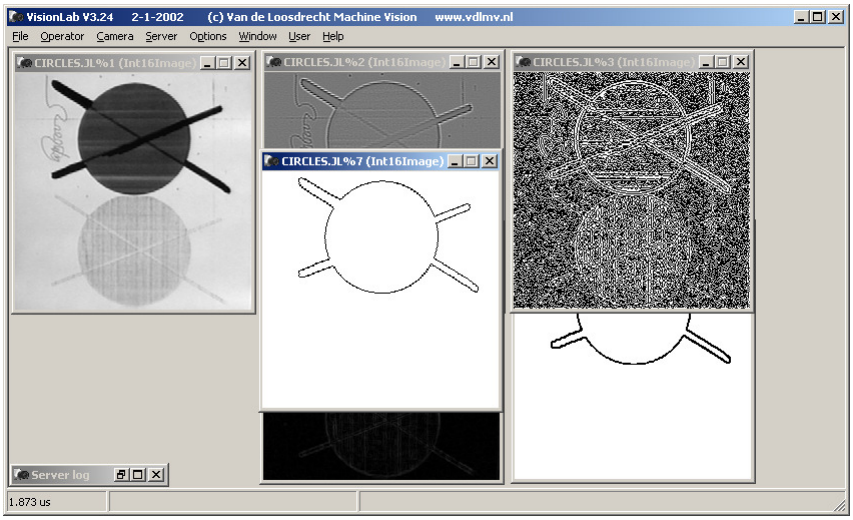
27-aug-18 Edge detection 34

Sobel **Threshold 200 10000**



27-aug-18 Edge detection 35

Multiply zero crossing with threshold



27-aug-18 Edge detection 36

Marr - Hildreth

MarrHildreth (srcImage, destImage, sigmaG, sigmaLoG, minEdge)

This operator calculates a binary image with the positions of the edges.

Algorithm:

- **First the Gaussian smoothing is performed (to 'connect' the edges)**
- **ZeroCrossings of the 2nd derivative (LoG) are multiplied with the high edges of the first derivative**

Parameters:

- **sigmaG: standard deviation for Gaussian smoothing**
- **sigmaLoG: standard deviation for the LoG operator**
- **minEdge: the minimal level for the first derivative**

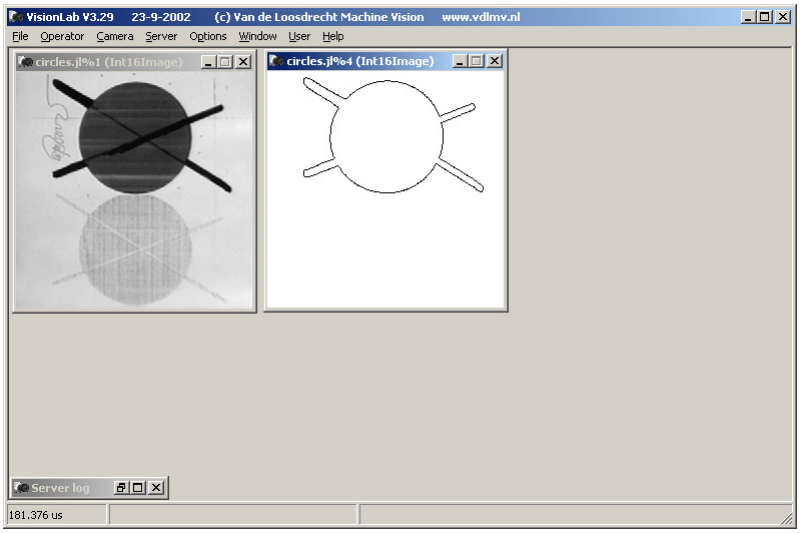
27-aug-18 Edge detection 37

Marr - Hildreth

- **Open image circles.jl**
- **MarrHildreth 0 1 200**

27-aug-18 Edge detection 38

MarrHildreth 0 1 200



27-aug-18 Edge detection 39

Canny (*)

Canny (srcImage, destImage, sigma, low, high, connected)

This operator calculates a binary image with the positions of the edges.

Algorithm and parameters:

- **Gaussian smoothing with *sigma***
- **Sobel edge detection**
- **Maxima of the edges magnitudes are searched for and linked. All pixels with a edge greater than *high* are selected as object pixels. These object pixels are used as seeds. All connected neighbours of the seeds with a edge greater than *low* are added to the object pixels. This growing process is repeated until no pixels are added.**

27-aug-18 Edge detection 40

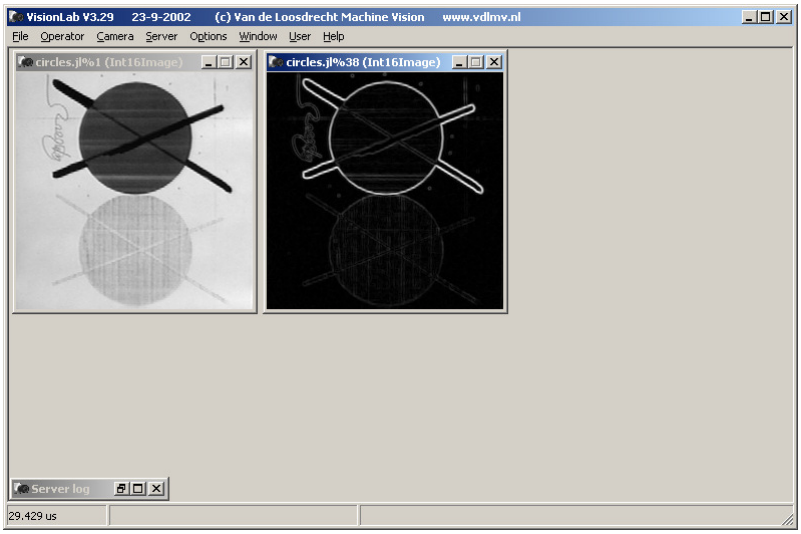
Demonstration Canny (*)

Try to find the signature and 'big circle'

- **Open image circels.jl**
- **Sobel GradientMagnitude 10000 0**
- **Threshold edge 60 1000, -> to much**
- **Threshold edge 150 1000, -> only 2 disconnected pixels of signature**
- **Canny image 0 60 150 EightConnected -> position of signature**

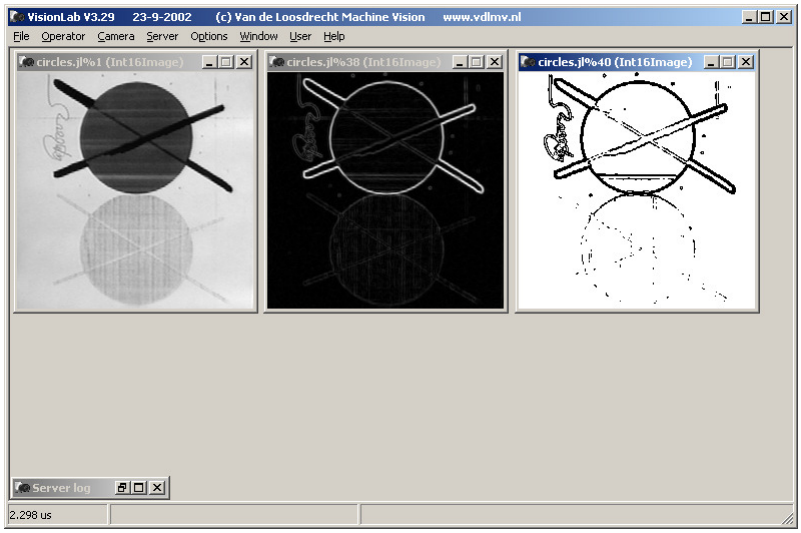
27-aug-18 Edge detection 41

Sobel GradientMagnitude 10000 0 (*)



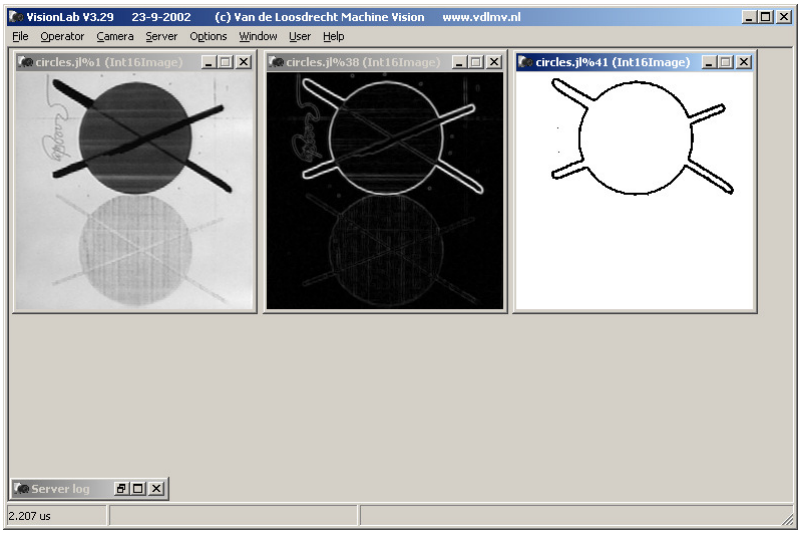
27-aug-18 Edge detection 42

Threshold edge 60 1000 (*)



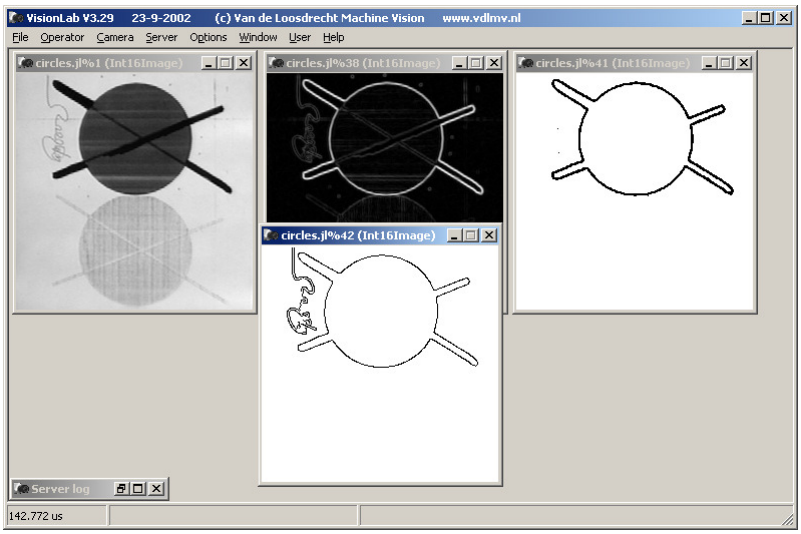
27-aug-18 Edge detection 43

Threshold edge 150 1000 (*)



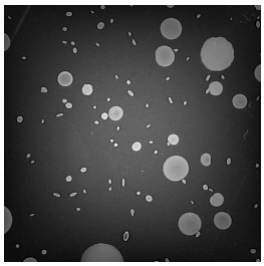
27-aug-18 Edge detection 44

Canny image 0 60 150 EightConnected (*)



27-aug-18 Edge detection 45

Exercise Segmentation using edge detection



- **Use image shading_c.jl in the excerise directory**
- **Try to find good threshold values in order to separate the cells from the background**
  This will be unsuccessful due to uneven lightning conditions
- **Use Sobel edge detection to find the borders of the cells and then to segment the image**

- **See shading_c_sobel.jls for answer**

27-aug-18 Edge detection 46

Exercise (*)

- **Experiment with the the other edge detection operators on image shading_c.jl**

FindEdgeLine

FindEdgeLine (image, middlePoint, endLine, endBox, lineDistance, outlierDistance, nrIterations)

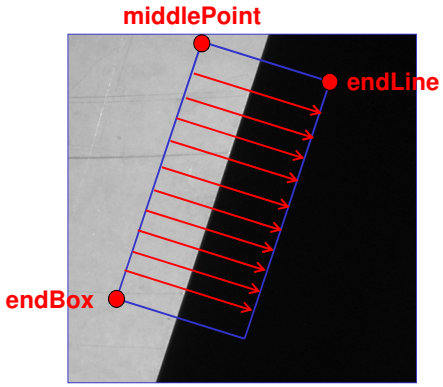
This operator finds with subpixel precision a line with the largest edges within the specified rectangle middlePoint, endLine and endBox.

In the specified rectangle scan lines will be tested at the specified lineDistance. The rectangle should have a width of at least 5 pixels.

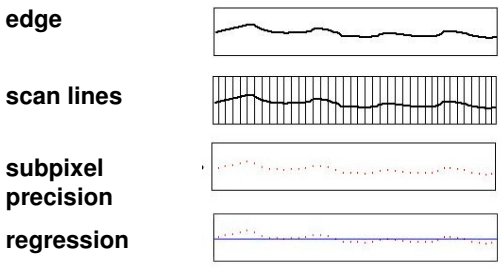
If outLayerDistance is greater then zero then the regression algorithm is repeated for nrIterations. In each next iterations only pixel with a distance smaller then outlierDistance to the previous found line are used in the calculation of the next line.

The function result is the start and the end coordinate of the line found and the number of pixels found on the line.

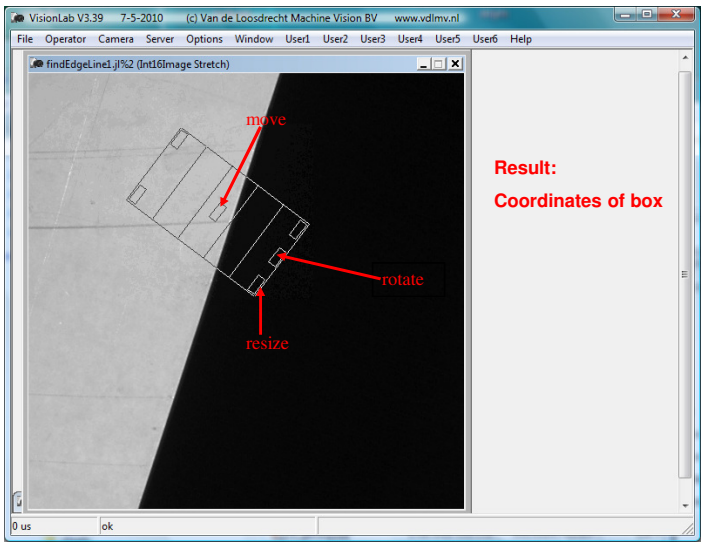
Specification of rectangle for scan lines



Finding the line



Raster tool



27-aug-18 Edge detection 51

Demo raster grid tool

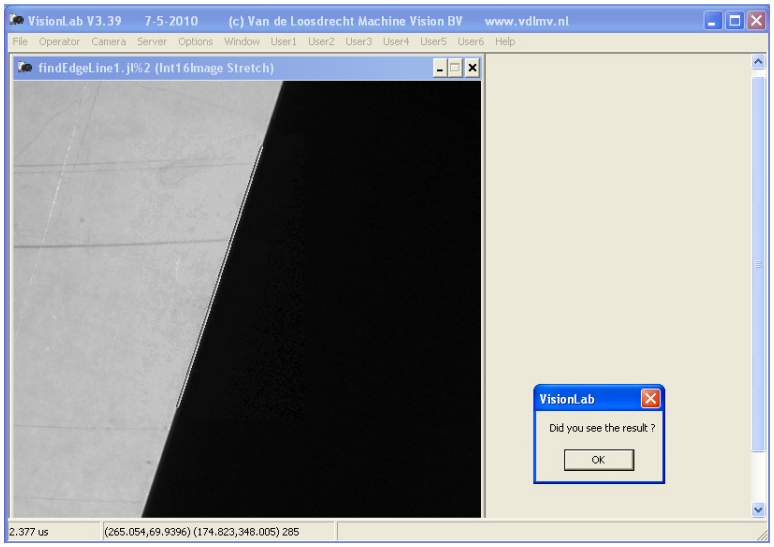
- **Open image findEdgeLine1.jl**
- **Open grid tool from menu Operator | Widget tools | FindEdgeLine**
- **Drag landmarks to position**
- **Select FindEdgeLine operator from menu Operator | EdgeDetection (line distance = 1, outlierDistance = 10, iteration = 10)**

27-aug-18 Edge detection 52

Drag landmarks to position

27-aug-18 Edge detection 53

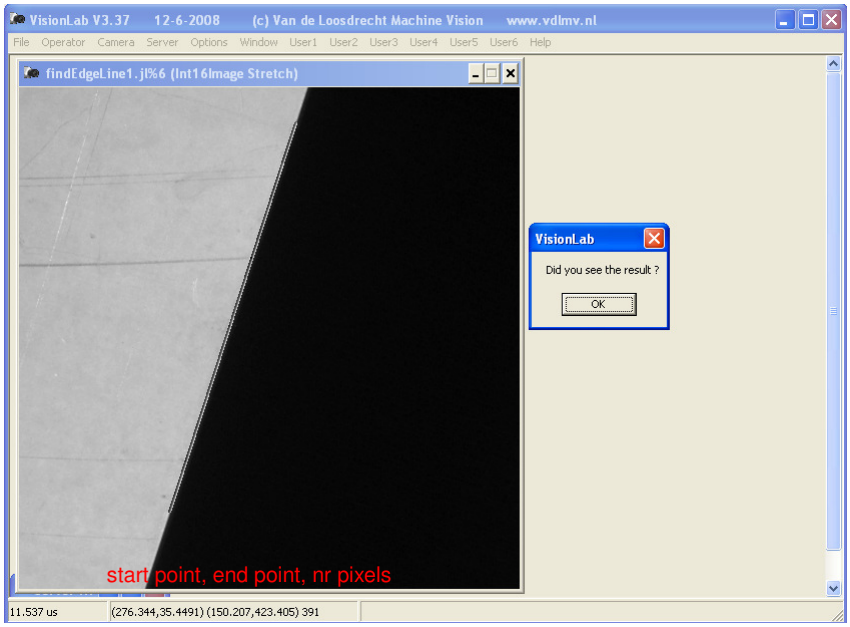
FindEdgeLine operator



27-aug-18 Edge detection 54

Demonstration FindEdgeLine

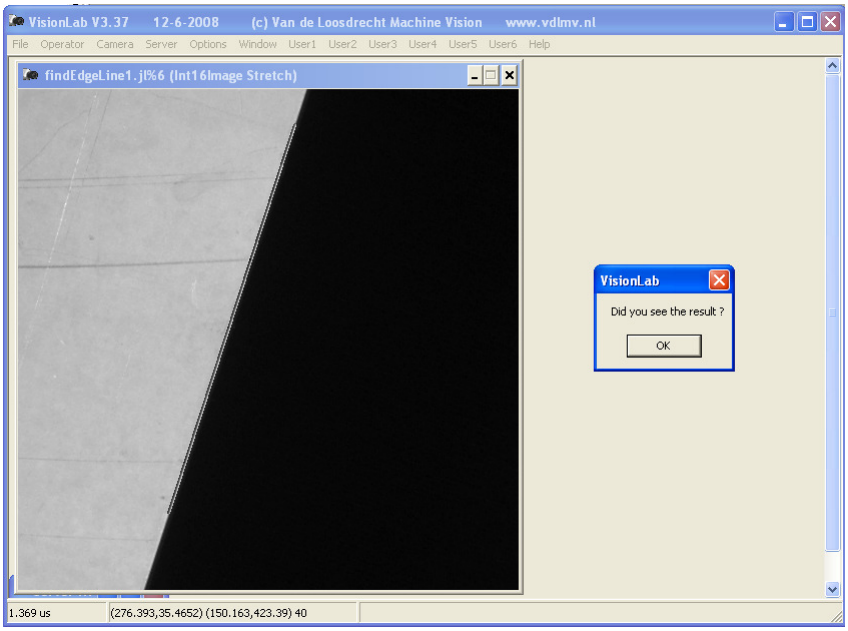
- **Open image findEdgeLine1.jl**
- **FindEdgeLine (200,10) (350,60) (80,400) 1 10 1**



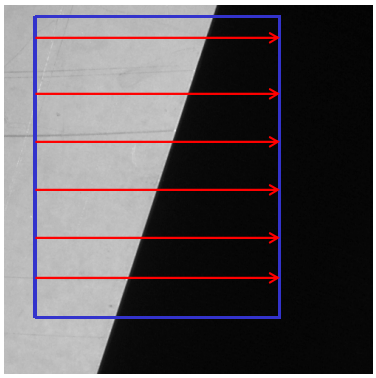
Demonstration FindEdgeLine

Increase speed by setting lineDistance to 10, decrease of accuracy (see number of points found)

- **FindEdgeLine (200,10) (350,60) (80,400) 10 10 1**

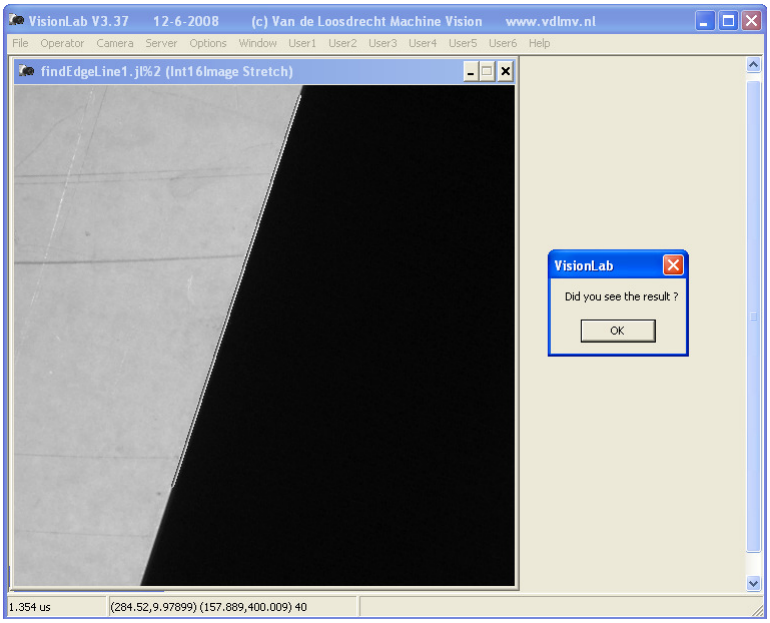


Example scan lines not perpendicular at edge



Demonstration scan lines not perpendicular to edge

- **Open image findEdgeLine1.jl**
- **FindEdgeLine (20,10) (350,10) (20,400) 1 10 1**



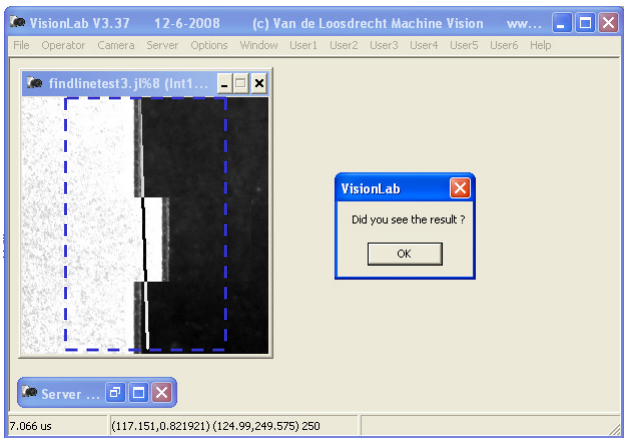
27-aug-18 Edge detection 61

Demonstration FindEdgeLine with outliers

- **Open image findedgeline2.jl**
- **FindEdgeLine (50,1) (200,1) (50,250) 1 0 1**

27-aug-18 Edge detection 62

Demonstration FindEdgeLine with outliers no iterations



27-aug-18 Edge detection 63

Demonstration FindEdgeLine with outliers

- **FindEdgeLine (50,1) (200,1) (50,250) 1 10 4**

27-aug-18 Edge detection 64

Demonstration FindEdgeLine with outliers
4 iterations

27-aug-18 Edge detection 65

FindSubEdgeOnLine

FindSubEdgeOnLine (image, start, end)

This operator finds with subpixel precision an edge on the line that lies between the given start and end coordinate. The start and end coordinate of this line can also be provided with subpixel precision. If multiple edges are found, the operator will return the edge with the highest magnitude.

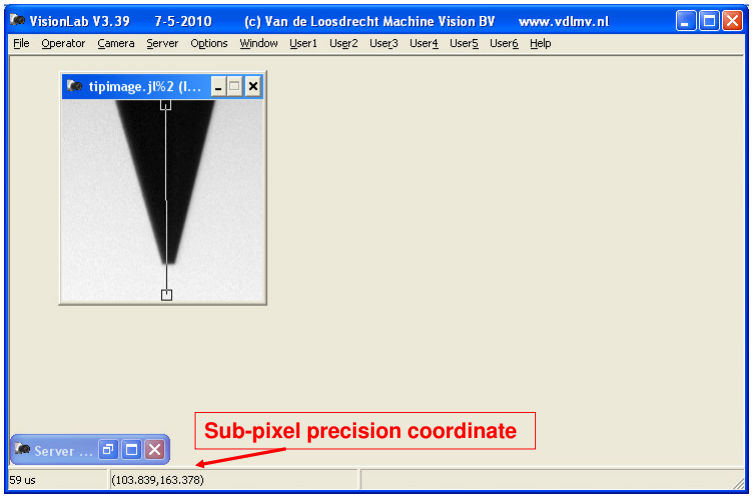
The function result is the coordinate with subpixel precision of the edge with highest magnitude on the line.

27-aug-18 Edge detection 66

Demonstration FindSubEdgeOnLine

- Open image tipimage.jl
- Open Line Tool widget: Operator | Widget Tools | FindSubEdge(s)OnLine
- Drag landmarks to position
- Apply operator FindSubEdgeOnLine

Demonstration FindSubEdgeOnLine



FindSubEdgesOnLine

FindSubEdgesOnLine (image, start, end, minEdge, edgeTab)

This operator finds with subpixel precision the all edges with an edge (difference between the 2 neighbours of a pixel) larger then minEdge on the specified line.

The function result is the number of edges found.

Demonstration FindSubEdgesOnLine

- **Open image tape.jl**
- **Open Line Tool widget: Operator | Widget Tools | FindSubEdge(s)OnLine**
- **Drag landmarks to position**
- **Apply operator FindSubEdgesOnLine, minEdge = 60**

Demonstration FindSubEdgesOnLine

27-aug-18 Edge detection 71

FindEdgeCircle

FindEdgeCircle (image, middlePoint, nrSamples, minR, maxR, outlierDistance, nrIterations)

This operator finds with subpixel precision a circle within the specified disk shape specified by middlePoint, minR and maxR. In the specified disk shape nrSamples scan lines tested starting from a distance minR from the middlepoint and ending at a distance maxR from the middlepoint. The probe lines will be equally divided in the space bounded by middlePoint, minR and maxR.

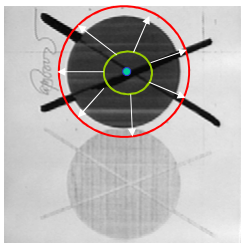
Note: the number of probe lines will be nrSamples rounded up to the next multiple of 4. maxR must be > minR + 5.

If outLayerDistance is greater then zero then the regression algorithm is repeated for nrIterations. In each next iterations only pixel with a distance smaller then outlierDistance to the previous found line are used in the calculation of the next line.

The function result is the center coordinate, the radius and the number of pixels found on the circle.

27-aug-18 Edge detection 72

Specification of disk shape for scanlines



middlePoint

minR

maxR

Scan lines

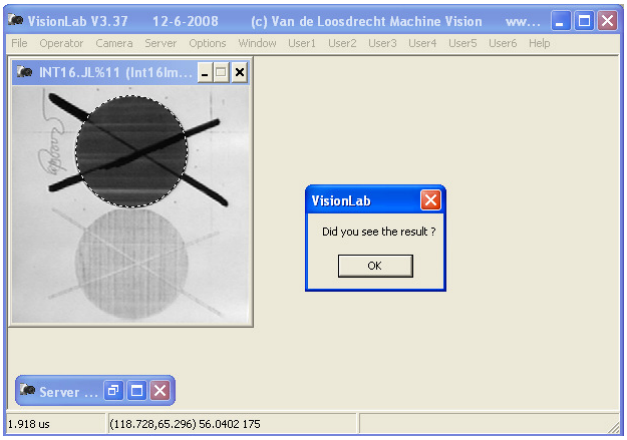
27-aug-18 Edge detection 73

Demonstration FindEdgeCircle (*)

- Open image circles.jl
- FindEdgeCircle (118,65) 200 20 65 1 1
- FindEdgeCircle (118,65) 50 20 65 1 1 (faster)

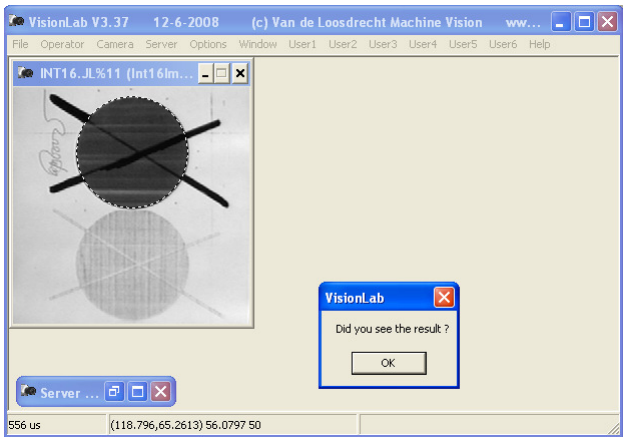
27-aug-18 Edge detection 74

FindEdgeCircle (118,65) 200 20 65 1 1 (*)

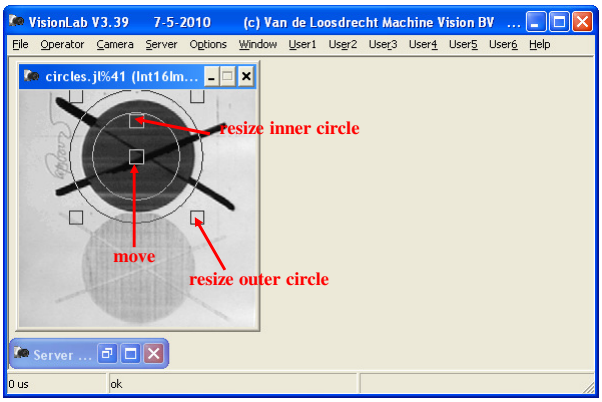


center, radius, nr pixels

FindEdgeCircle (118,65) 50 20 65 1 1 (*)



FindEdgeCircle with MinMaxCircleTool widget



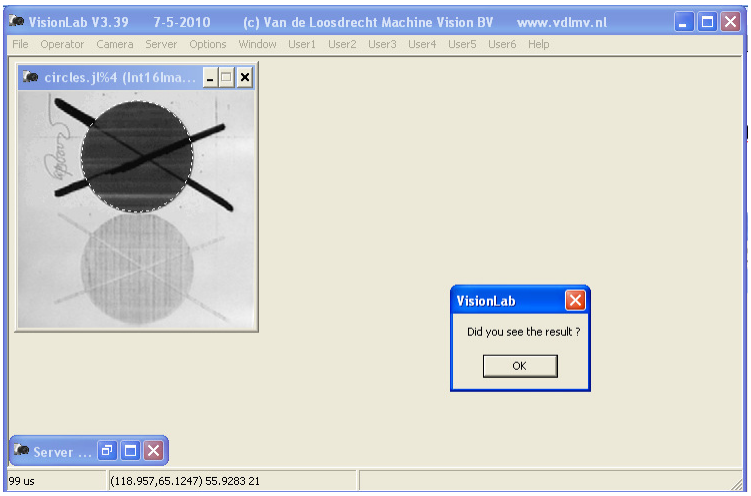
27-aug-18 Edge detection 77

FindEdgeCircle with MinMaxCircleTool widget

- **Open min max circles tool from menu Operator | Widget tools | FindEdgeCircle**
- **Drag landmarks to desired position**
- **Select FindEdgeCircle operator from menu Operator | Transforms**

27-aug-18 Edge detection 78

FindEdgeCircle with MinMaxCircleTool widget



27-aug-18 Edge detection 79

Alternative for finding lines and circles

Alternative operators to find lines and circles are based on the Hough transform, see the chapter about Hough transforms

Edge based:

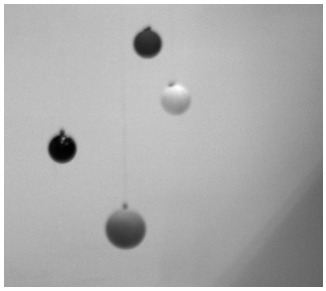
- **Fast**
- **Search area must contain *only* edges to find**
- **Can find only 1 line or circle**
- **Outliers cause problems**

Hough based:

- **Slower**
- **Search area can be whole image**
- **Can find more then 1 lines or circles**
- **Less problems with outliers**

27-aug-18 Edge detection 80

Exercise search balls using FindEdgeCircle



- Use robot_balls.jl in the exercise directory
- Write a script to find the four balls
- Hint: first find center of candidate blobs using BlobAnalysis
- See find_edge_balls.jls for answer

27-aug-18 Edge detection 81

FindFirstTransitions (*)

FindFirstTransitions (srcImage, destImage, threshold, viewPoint, objectBrightness)

Finds the first transitions from a specified viewPoint. The destination image is initialized with the pixel values containing the distance from the border of the image to the occurrence of the first transitions
Note: the destination image is one dimensional

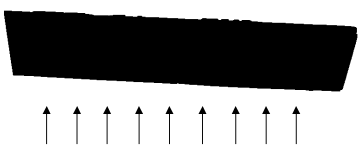
Parameters:
- **threshold: threshold value of the transition**
- **viewPoint: Direction from where to find the first transitions: Top, Left, Right or Bottom**
- **objectBrightness:**
  - **DarkObject: a transition to a lower value then threshold is considered as a transition**
  - **BrightObject: a transition to a higher value then threshold is considered as a transition**

27-aug-18 Edge detection 82

FindFirstTransitions Example (*)

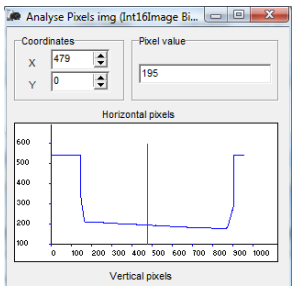
- Open image nummerbord_color.jl
- Segmentate image with ThresholdRGBchannels Int16Image 185 243 107 191 0 7
- FillHoles FourConnected
- RemoveBlobs Area 1 1000 UseX
- FindFirstTransitions 1 Bottom BrightObject (in Point menu)

FindFirstTransitions Example (*)



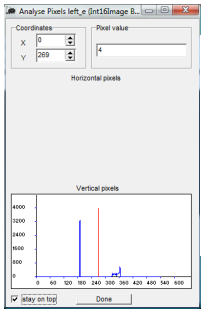
FindFirstTransition

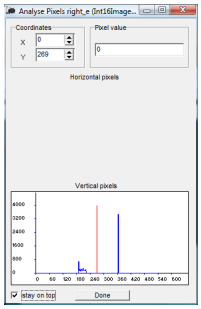
Src dst 1 Bottom BrightObject



Corner detection using FindFirstTransitions (*)

1) FindFirstTransitions in every direction
2) Edge detection
3) High peaks are sharp edges





Left Right

Corner detection using FindFirstTransitions (*)

Script: Edge_corner1.jls (Alternative: Edge_corner2.jls)

FindFirstEdges (*)

FindFirstEdges (srcImage, dstImage, threshold, viewPoint)

Finds the first edge from a specified viewPoint. The destination image is initialized with pixel values containing the distance from the border of the image to the occurrence of the first edge
Note: the destination image is one dimensional

This operator uses the Sobel edge detection.

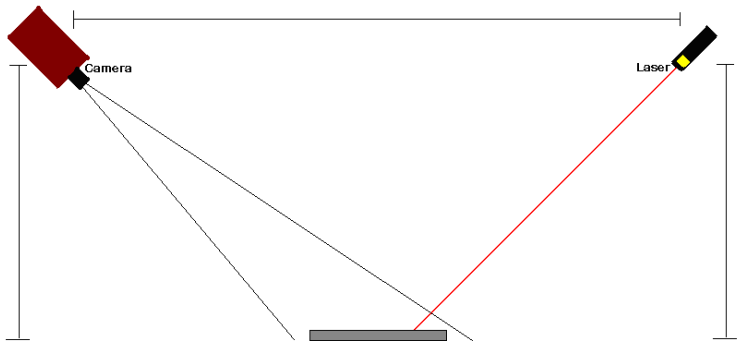
Parameters:

- **threshold: threshold value of the edge. An edge above this value is considered an edge**
- **viewPoint: Direction from where to find the first edge: Top, Left, Right or Bottom**

27-aug-18 Edge detection 87

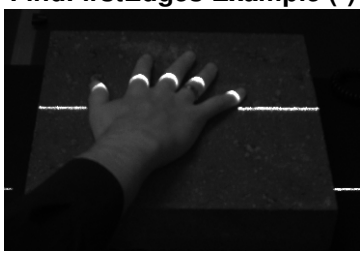
FindFirstEdges Example (*)

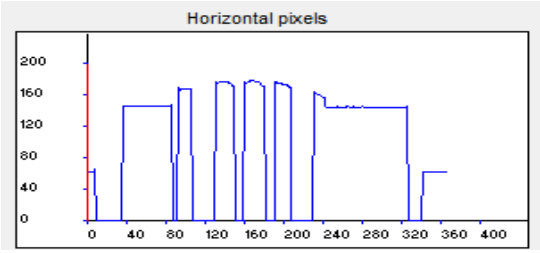
Laser triangulation (Generate 3D profile):



27-aug-18 Edge detection 88

FindFirstEdges Example (*)





27-aug-18 Edge detection 89

FindFirstEdges Example (*)

Scrip: find_laser.jls

```
lread img laser.jl
display img
$h = getheight img
FindFirstEdges img distance 50 Bottom
// correct the max values to zero
setselectedtovalue distance $h $h 0
display distance
```

27-aug-18 Edge detection 90