NHL STENDEN
computer vision & data science

Computer Vision

Distance and Hough Transforms

10 April 2018

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

Distance and Hough Transforms

Overview:

- **Distance transform**
  - **Pixel distance**
  - **Exact distance**
  - **Vector distance (*)**
  - **Euclidean distance (*)**
- **Hough transform**
  - **Circle**
  - **Line**

Distance transform

DistanceT srcImage destImage connected

This operator works on Binary Image (= source) and initialises a greyscale image (= destination).

Each pixel in an object is assigned a pixel value equal to its (eight or four connected) distance to the nearest background pixel.

Usage:

- **Fast alternative for repetitive erosion**

27-aug-18 Distance and Hough Transforms 3

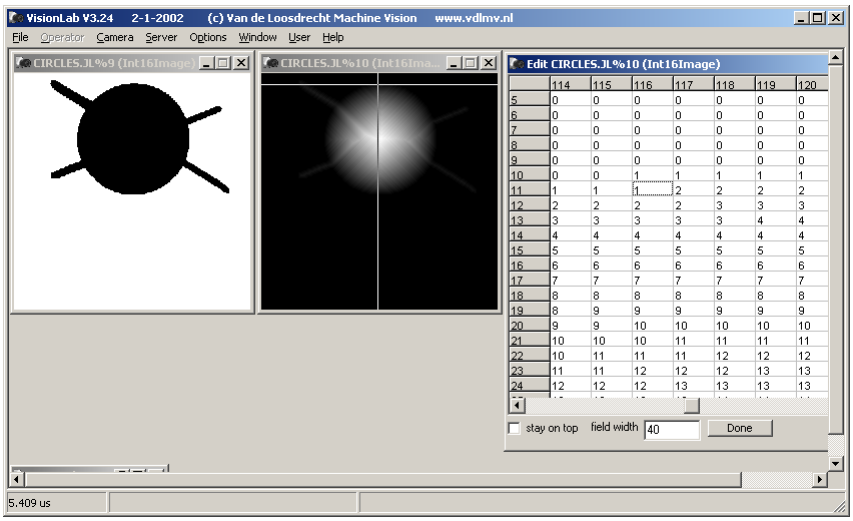
Demonstration Distance Transform

- **Open image circles.jl**
- **Threshold 0 100**
- **Distance transform EightConnected**
  - **Analyse image with Edit (pixel 116,11 = 1)**
- **Distance transform FourConnected**
  - **Analyse image with Edit (pixel 116,11 = 2)**

- **Fast alternative for repetitive erosion:**
- **Use script dt_speed.jls, *use single step mode to show timer results***
- **Apply threshold 5 100 on EightConnected distance transform**
- **Apply 4 x Erosion with full 3x3 mask on thresholded image, note time needed for operations**

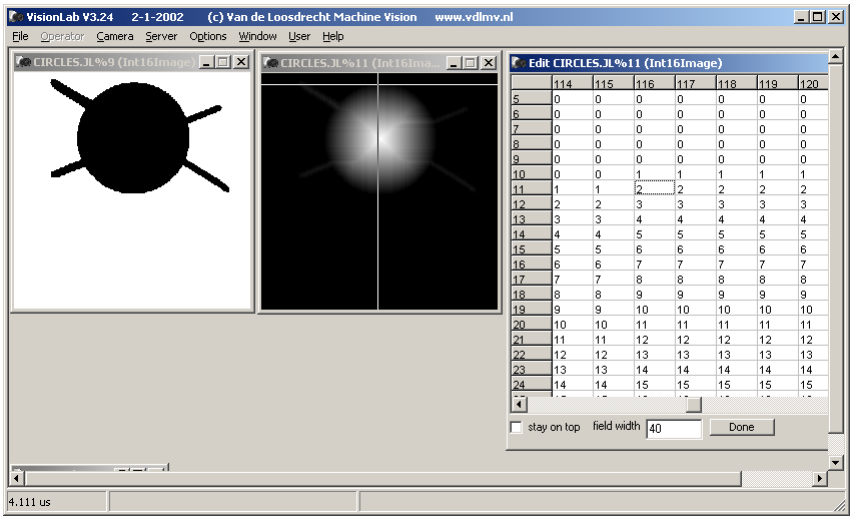
27-aug-18 Distance and Hough Transforms 4

Distance transform EightConnected



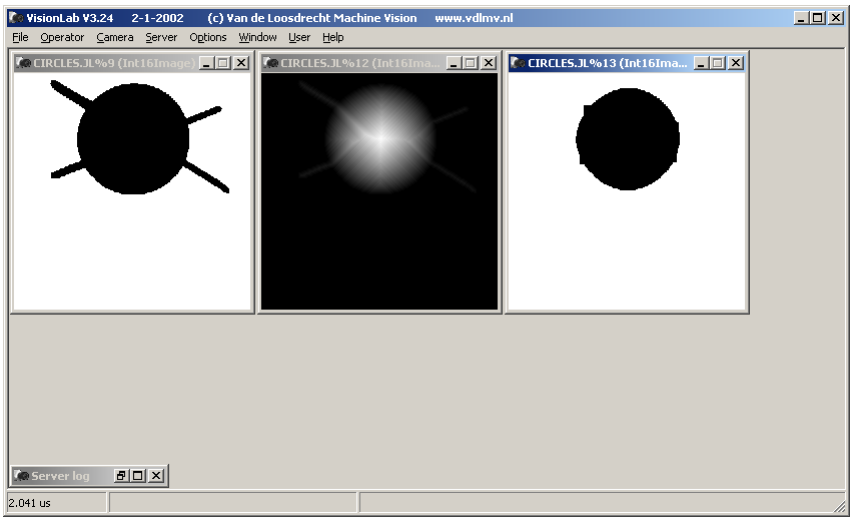
27-aug-18 Distance and Hough Transforms 5

Distance transform FourConnected



27-aug-18 Distance and Hough Transforms 6

Fast alternative for repetitive erosion:
Threshold 5 100 on EightConnected distance transform



27-aug-18 Distance and Hough Transforms 7

Exact Distance transform

ExactDistanceT image precision

This operator works on binary image and converts it to a greyscale integer image.

Each pixel in an object is assigned a pixel value equal to its exact distance from the nearest background pixel or border of the image.

The distances are calculated with floating point precision. The final result for each distance is multiplied with the precision factor (2nd parameter) and converted to an integer value.

Usage:

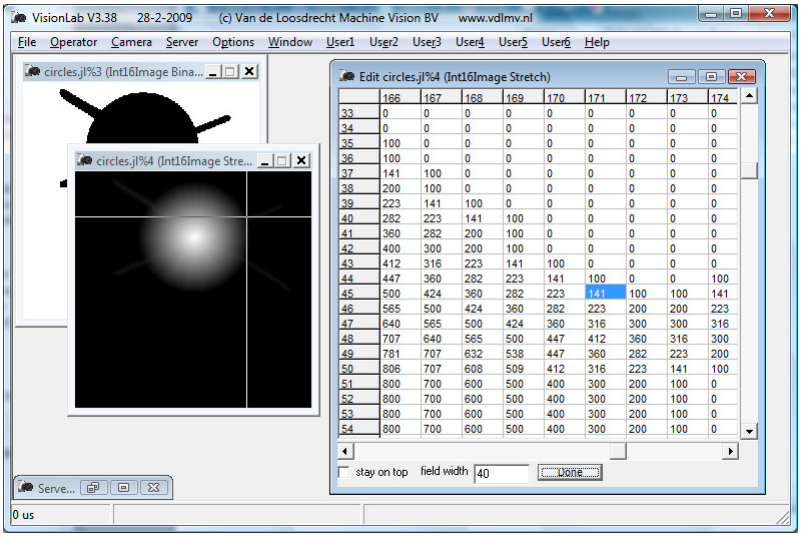
- **Fast alternative for repetitive erosion**

27-aug-18 Distance and Hough Transforms 8

Demonstration Exact Distance Transform

- **Open image circles.jl**
- **Threshold 0 100**
- **ExactDistanceT 100**
- **Explain scaling**

Demonstration Exact Distance Transform



Vector Distance transform (*)

ExactDistanceT srcImage destImage

This operator works on binary image (= source) and initialises a complex float image (= destination).

Each pixel in an object is assigned a pixel value equal to the vector with the smallest distance from the nearest background pixel or border of the image.

The vector is represented as a complex pixel.

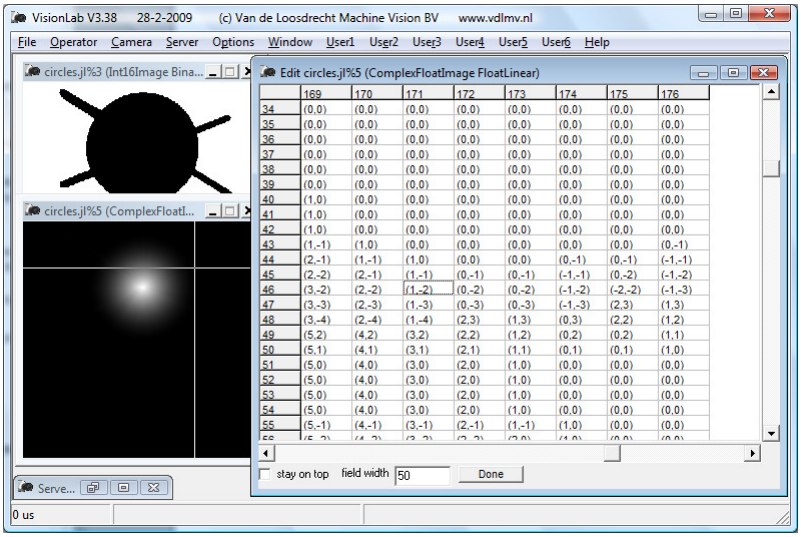
27-aug-18 Distance and Hough Transforms 11

Demonstration Vector Distance Transform (*)

- **Open image circles.jl**
- **Threshold 0 100**
- **VectorDistanceT 100**

27-aug-18 Distance and Hough Transforms 12

Demonstration Vector Distance Transform (*)



Euclidean Distance Transform (*)

EuclideanDistanceT srcImage destImage mask result precision ydivx

This operator works on Binary Image (= source) and initialises a greyscale image (= destination).

Each pixel in an object is assigned a pixel value equal to its Euclidean distance to the nearest background pixel.

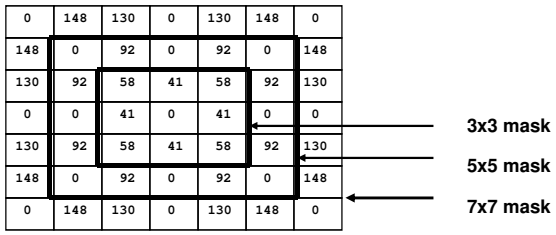
Implementation restriction:
blobs should be at least n pixels distance from the border of the image. (mask3x3: n=1, mask5x5: n=2, mask7x7: n=3)

Euclidean Distance Transform (*)

Implementation notes:

- **calculations in "41 base number", 58/41 = √2**
- **accuracy can be increased by using bigger mask**
- **ydivx gives the ratio of the pixelsize = 1 for square pixels**

Euclidean Distance Transform (*)



- **Masks for square pixels and base number 41**
- **Maximum blob diameter for Int16Image: (2^{15}/41) * 2 + 1 = 1599 pixels**

Euclidean Distance Transform (*)

0	148	130	0	130	148	0
148	0	92	0	92	0	148
130	92	58	41	58	92	130
0	0	41	0	41	0	0
130	92	58	41	58	92	130
148	0	92	0	92	0	148
0	148	130	0	130	148	0

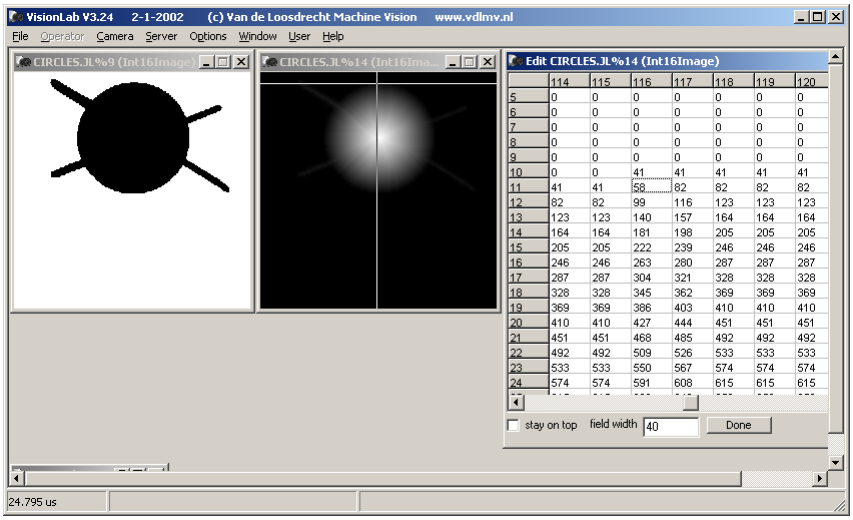
3x3 mask
5x5 mask
7x7 mask

- **Sensitive to rotation of the object, maximum errors: mask3x3 = 7.9%, mask5x5 = 2.5% and mask7x7 = 1.2%**
- **Example: distance knight move with mask3x3: 58 + 41 = 99. Error: (99/41) / sqrt(5) = 1.079**

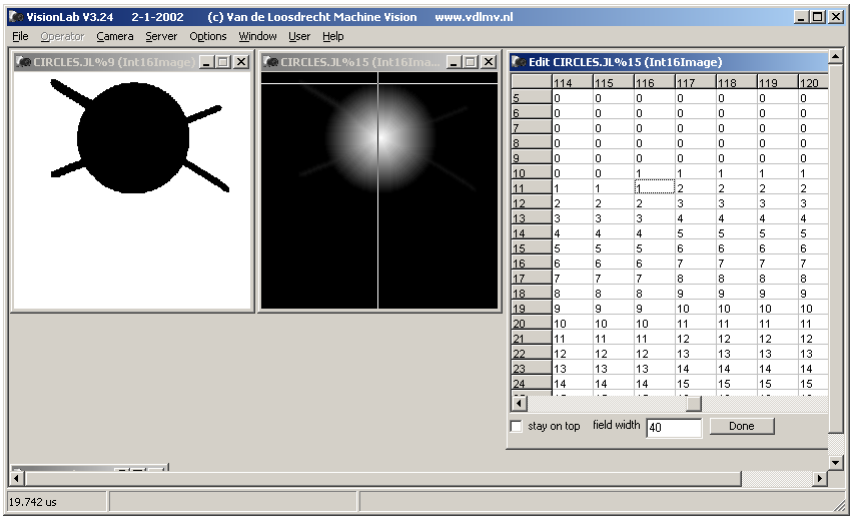
Demonstration Euclidean Distance Transform (*)

- **Open image circles.jl**
- **Threshold 0 100**
- **(EuclideanDistanceT EDTMask3x3 NoScaleEDT 41 1 (*))**
- **EuclideanDistanceT EDTMask3x3 ScaleEDT 41 1**
- **(Explain scaling (*))**

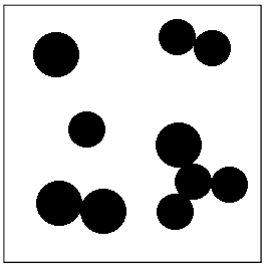
EuclideanDistanceT NoScaleEDT (*)



EuclideanDistanceT ScaleEDT (*)



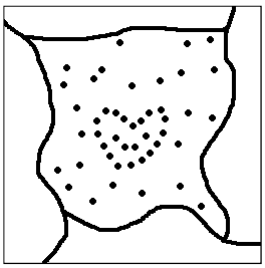
Exercise 1 Distance Transform



- **Use image connectedballs.jl**
- **Calculate the number of balls in the image**

- **answer: connectedballs.jls**

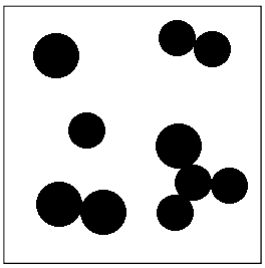
Exercise 2 Exact Distance Transform (*)



- **Use image city.jl**
- **Calculate the average shortest distance from the centre of the cities to the highway**

- **answer: city.jls**

Exercise 3 Exact Distance Transform (*)



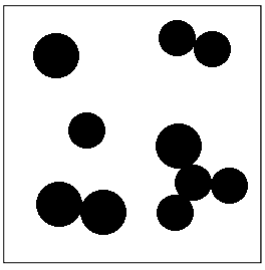
- **Use image connectedballs.jl**
- **Calculate "the watershed" between the balls**

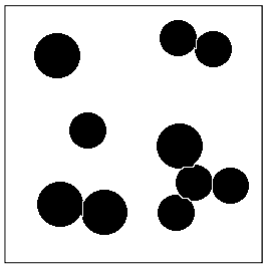
- **answers: watershed_conballs2.jls , watershed_conballs.jls and watershed_conballs3.jls**
  Note: alternative solution is using "SeparateBlobs EightConnected 20"

27-aug-18 Distance and Hough Transforms 23

Exercise 4 Exact Distance Transform (*)

- **Implement script for SeparateBlobs using Watershed**
- **Test on image connectedballs.jl**





Answer: SeparateBlobs_with_Watershed.jls

27/08/2018 Segmentation 24

Hough Transform

- **Hough Circle Transform**
- **Hough Line Transform**

Hough Circle Transform

Task: find the best match for a circle

Image space

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

a,b,R: known

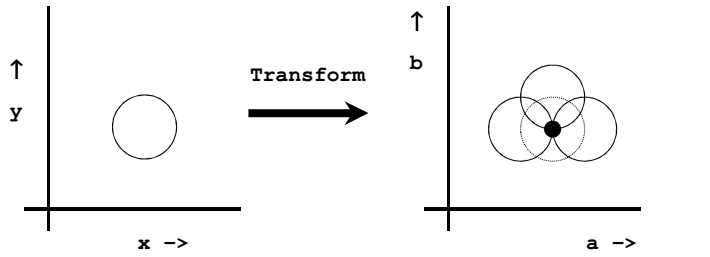
x,y: unknown

Hough space

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

x,y: known

a,b: unknown, R: fixed



Hough Circle Transform

Algorithm:

- **Initialise Hough space to 0**
- **For all object pixels 'draw' in Hough space a circle with radius R by incrementing all pixels in circle.**
- **Find maximum for position centre**

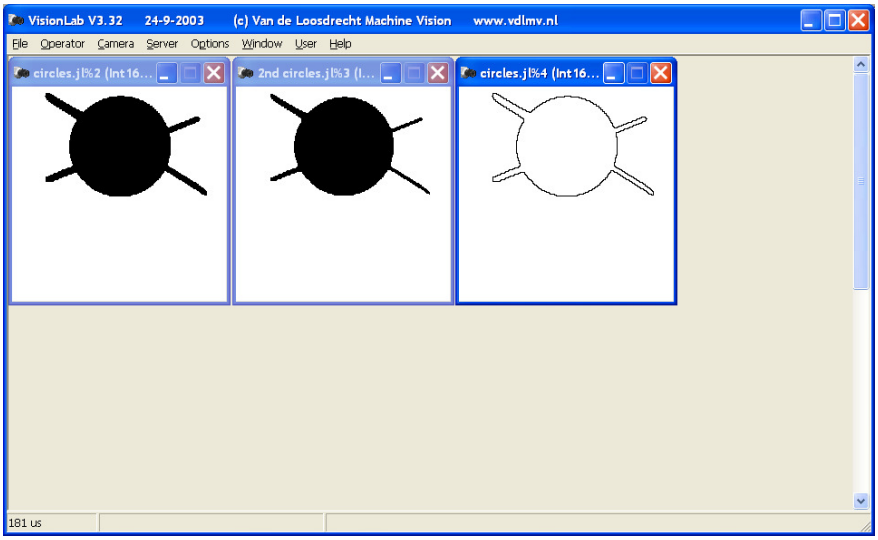
This can be repeated for all radii R to be searched, maximum over all Hough spaces will give the answer

Demonstration Hough Circle Transform

- **(note: log mode should be normal, not CSV)**
- **Open image circles.jl**
- **Threshold 0 100**
- **Erosion with full 3x3 mask**
- **Subtract eroded image from thresholded image in order to get a border image**
- **Hough circle transform on border image with radius 55**
- **(use gamma correction in point menu with factor 0.25 for better contrast in display with beamer)**
- **Analyse result with edit pixels**
- **Use Hough best circle with radius 55 to find position of circle**

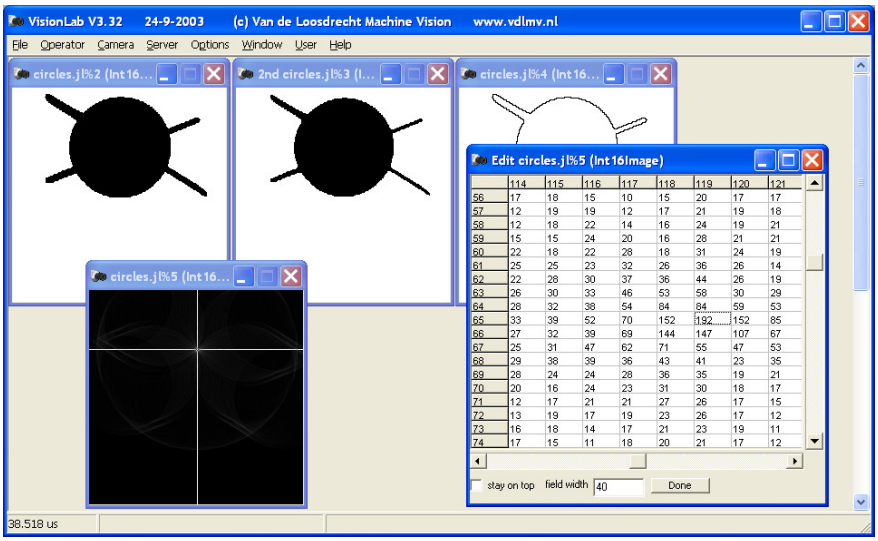
- **An unknown radius can be found with**
  FindBestCircle borderimage 50 58 0.1
  (result = centre of gravity radius nrofhits)
  For animation use script HoughAnimation.jls

Threshold **Erosion** **Border**



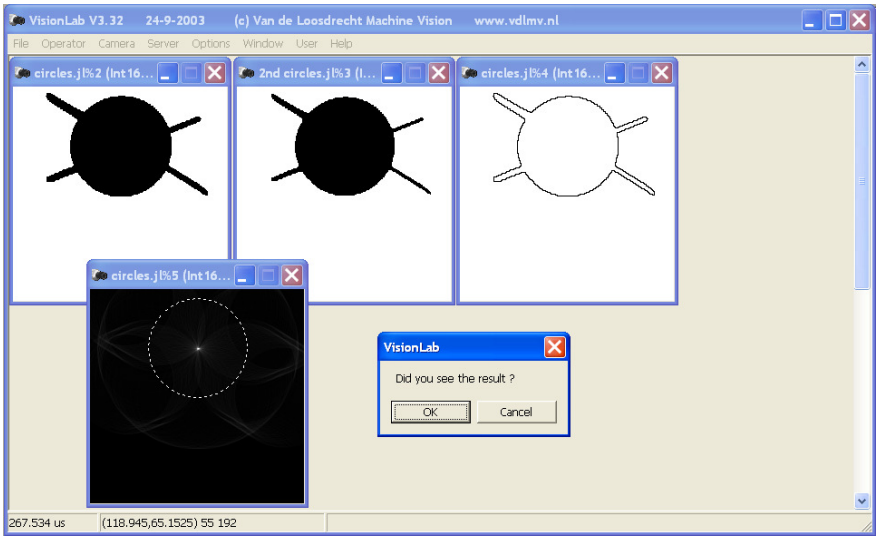
27-aug-18 Distance and Hough Transforms 29

Hough Circle Transform with radius 55



27-aug-18 Distance and Hough Transforms 30

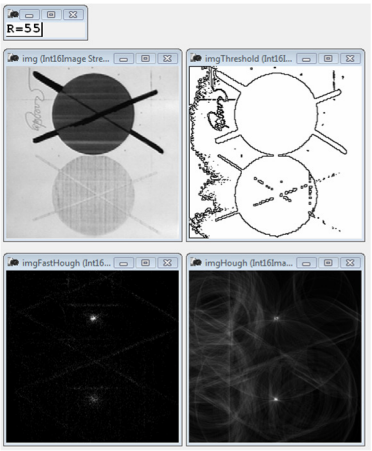
Hough Best Circle with radius 55 to find position of circle



27-aug-18 Distance and Hough Transforms 31

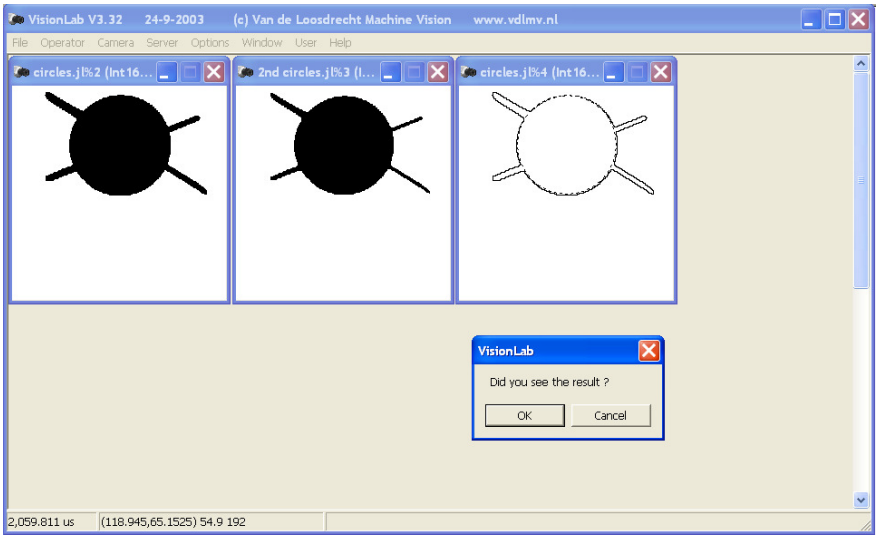
Animation Hough Circle Transform

- **Open script HoughAnimation.jls**



27-aug-18 Distance and Hough Transforms 32

Find Best Circle 50 58 0.1



27-aug-18 Distance and Hough Transforms 33

Find Best Circle

findbestcircle imageName minR maxR deltaR

findbestcircle is intended for a binary image

The operator searches in the image for the best match for a circle with minimum radius minR and maximum radius maxR and with a resolution of deltaR pixel.

**The operator will give as result the following string:
(centre co-ordinate) radius numberOfHits.**

Note sub-pixel precision.

27-aug-18 Distance and Hough Transforms 34

Find Fast Best Circle

findfastbestcircle imageName brightness edgeMin minR maxR deltaR

findfastbestcircle is intended for a grayscale image.

findfastbestcircle has two extra parameters:

- **brightness: determines whether the circle is dark or light or can be both relative to the background.**
- **edgeMin: the border of the circle is found using the Sharr edge detection method, all edge with a magnitude higher than edgMin are considered as candidate for the circle.**

Note sub-pixel precision.

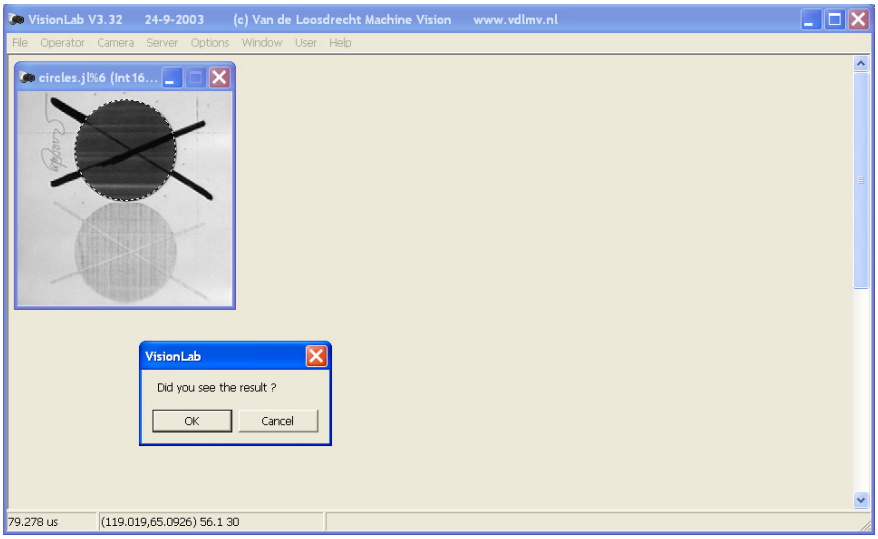
27-aug-18 Distance and Hough Transforms 35

Demonstration Find Fast Best Circle

- **Open image circles.jl**
- **findfastbestcircle DarkOrBrightCircle 500 50 58 0.1**

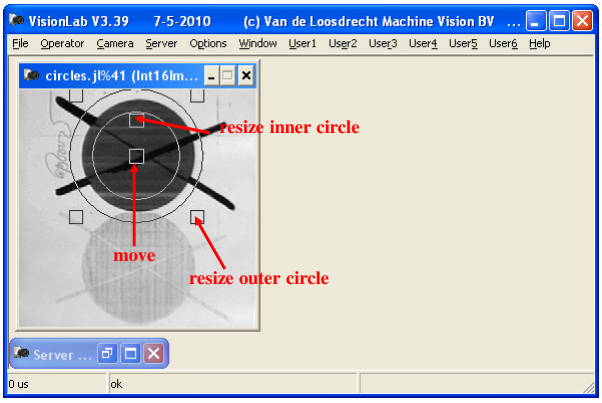
27-aug-18 Distance and Hough Transforms 36

Demonstration Find Fast Best Circle



27-aug-18 Distance and Hough Transforms 37

Find Fast Best Circle using widget FindEdgeCircleTool

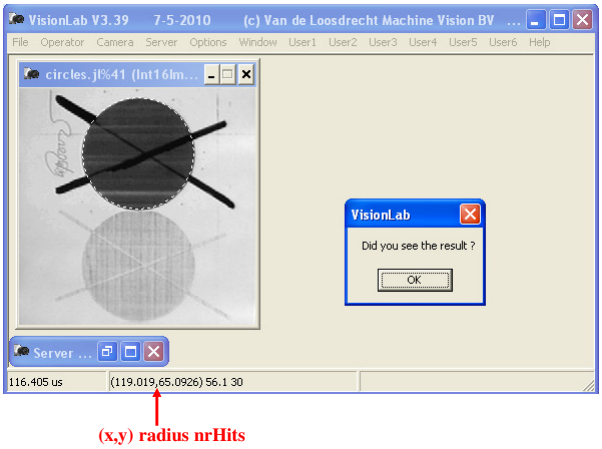


27-aug-18 Distance and Hough Transforms 38

Demo Find Fast Best Circle using widget FindEdgeCircleTool

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

Demo Find Fast Best Circle using widget FindEdgeCircleTool



Implementation FindFastCircle

- **The border of the circle is searched with an edge detection opeator**
- **For each border pixel the magnitude and direction of the edge is calculated**
- **With the direction the tangent of the circle is calculated**
- **The position of the center of the circle can be calculated**
- **For each candidate pixel only one point (the center) is 'drawn' in the Hough space**

Hough Circle Transform

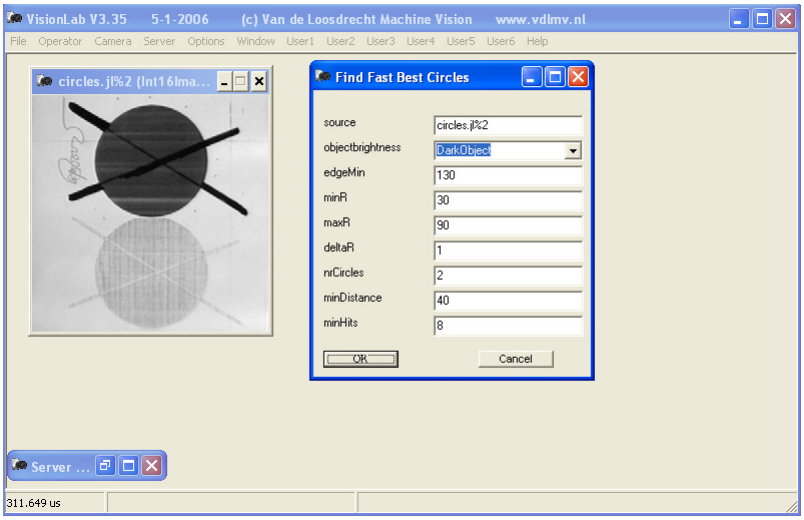
Explain difference in found circle radius:

- **FindBestCircle: r = 54.9**
- **FindFastBestCircle: r = 56.1**

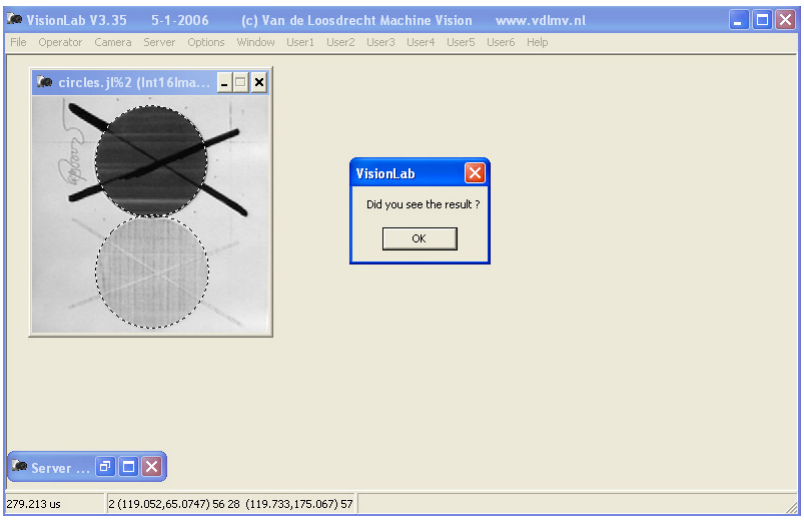
Demonstration Find Fast Best Circles

- **Open image circles.jl**
- **findfastbestcircles DarkOrBirghtCircle 130 30 90 1 2 40 8**

Find Fast Best Circles



Find Fast Best Circles



27-aug-18 Distance and Hough Transforms 45

Hough Line Transform

Task: find the best match for a line

Image space:

$y = a * x + b$

a,b known

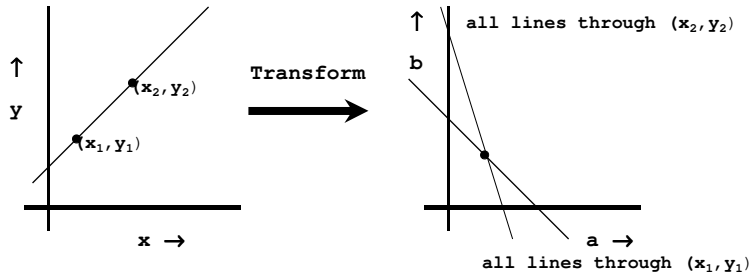
x,y unknown

Hough space:

$b = y - a * x$

x,y known

a,b unknown



27-aug-18 Distance and Hough Transforms 46

Hough Line Transform (*)

Task: find the best match for a line

Algorithm:

- **Initialise Hough space to 0**
- **For all object pixels 'draw' in Hough space all lines (draw = increment pixel value in Hough space)**
- **Maximum in Hough space gives a and b for best match**

Problem:

- **If line is vertical then a = infinite**

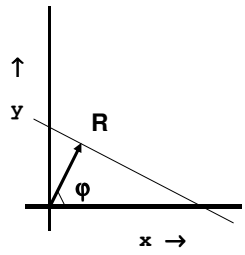
Solution:

- **Represent line in Hough space by polar co-ordinates: $x * \cos\varphi + y * \sin\varphi = R$**

Polar co-ordinates (*)

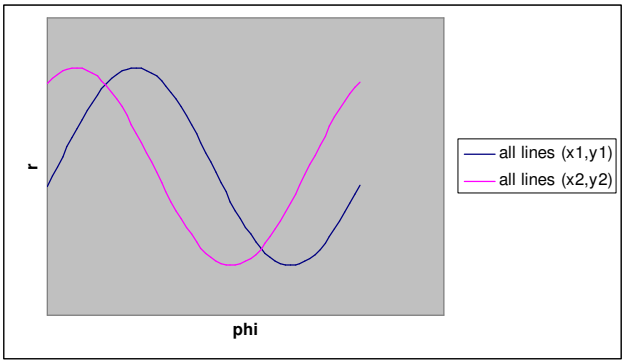
A line is represented by a normal vector with:

- **length R**
- **angle φ**



Polar co-ordinates (*)

Hough space of all lines through two points



27-aug-18 Distance and Hough Transforms 49

Find Best Line

Task: find the best match for a line

findbestline srcName minR maxR deltaR minPhi maxPhi deltaPhi

findbestline is intended for a binary image

This operator searches in the image for the best match for a line. The normal vector of this line (r,phi) and the number of hits are returned as result.

This line is searched for in the area of the image with the following limitations (in polar co-ordinates): r in [minR..maxR] and phi in [minPhi..maxPhi] in degress. Limitation of phi: -pi/2 <= phi <= pi, due to the fact that a normal vector can not be in the fourth quadrant.

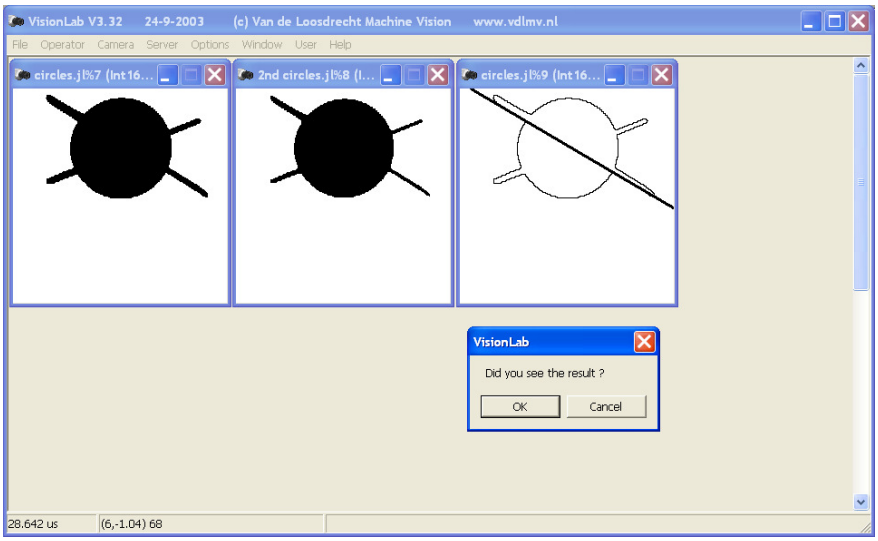
The resolution of the search is determined by deltaR and deltaPhi.

27-aug-18 Distance and Hough Transforms 50

Demonstration Hough Line Transform

- **Open image circles.jl**
- **Threshold 0 100**
- **Erosion with full 3x3 mask**
- **Subtract eroded image from thresholded image in order to get a border image**
- **findbestline borderimage 0 200 1 -1.5 3.14 0.01**

findbestline 0 200 1 -1.5 3.14 0.01



Find Fast Best Line

findfastbestline srcName minR maxR deltaR minPhi maxPhi deltaPhi edgeMin

findfastbestline is intended for a grayscale image.

findfastbestline has an extra parameter:

- **edgeMin: the border of the line is found using the Sharr edge detection method, all edges with a magnitude higher than edgMin are considered as candidates for the line.**

Implemention with an edge detection operator which finds the orientation of the line segments. For each pixel in the image only one pixel in the Hough space is incremented

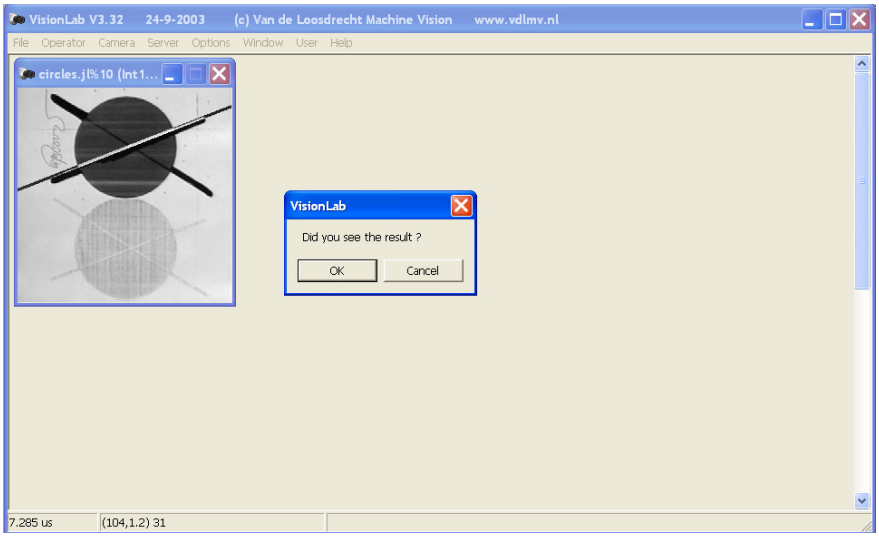
27-aug-18 Distance and Hough Transforms 53

Demonstration Find Fast Best Line

- **Open image circles.jl**
- **findfastbestline 400 0 200 1 -1.5 3.14 0.1**

27-aug-18 Distance and Hough Transforms 54

Demonstration Find Fast Best Line



27-aug-18 Distance and Hough Transforms 55

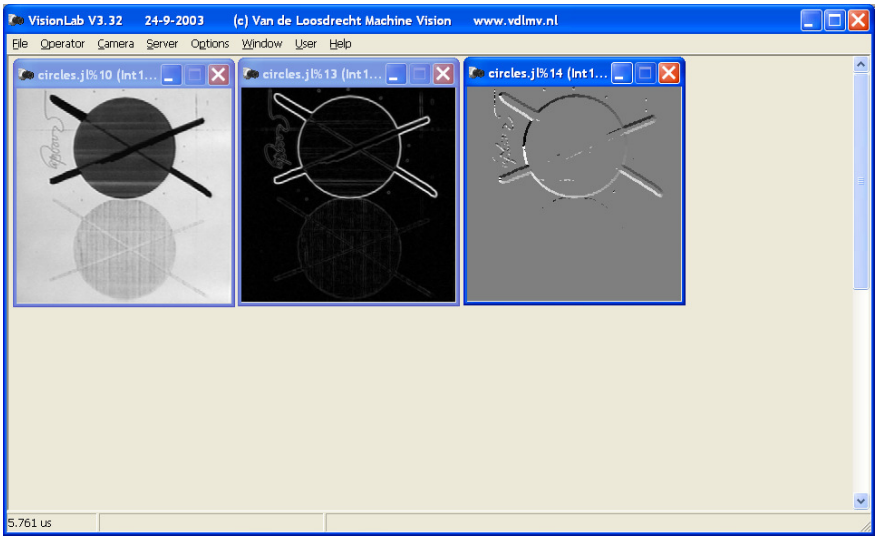
Find (Fast) Best Line (*)

Question:

Why is the result from Find Fast Best Line different from Find Best Line ?

27-aug-18 Distance and Hough Transforms 56

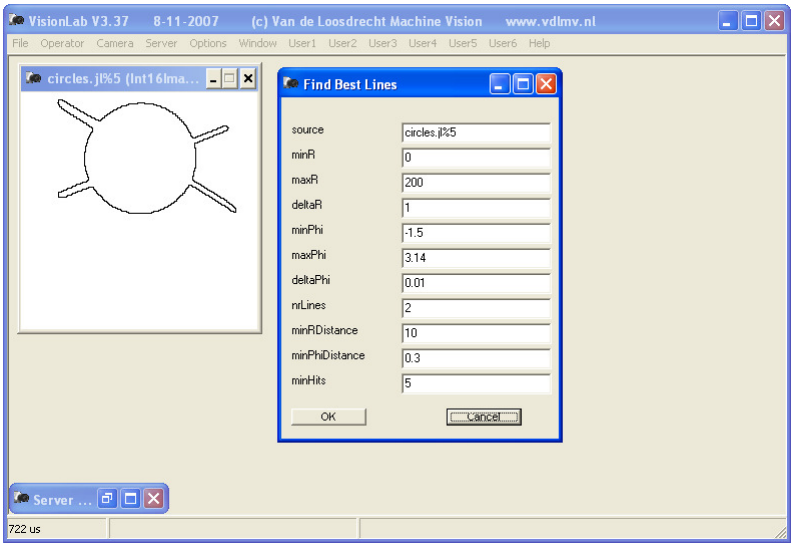
Scharr with minEdge = 400 (*)



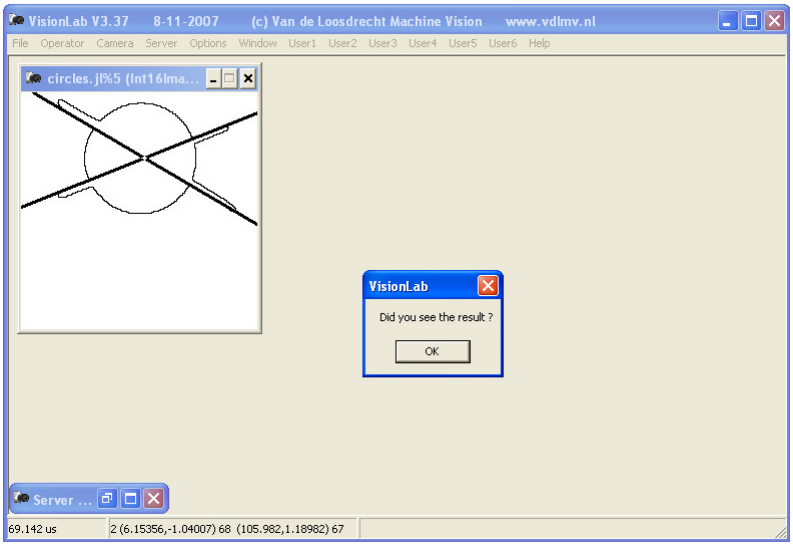
Demonstration Find Best Lines (*)

- **Open image circles.jl**
- **Threshold 0 100**
- **Erosion with full 3x3 mask**
- **Subtract eroded image from thresholded image in order to get a border image**
- **findbestlines borderimage 0 200 1 -1.5 3.14 0.01 2 10 0.3 5**

Find Best Lines (*)



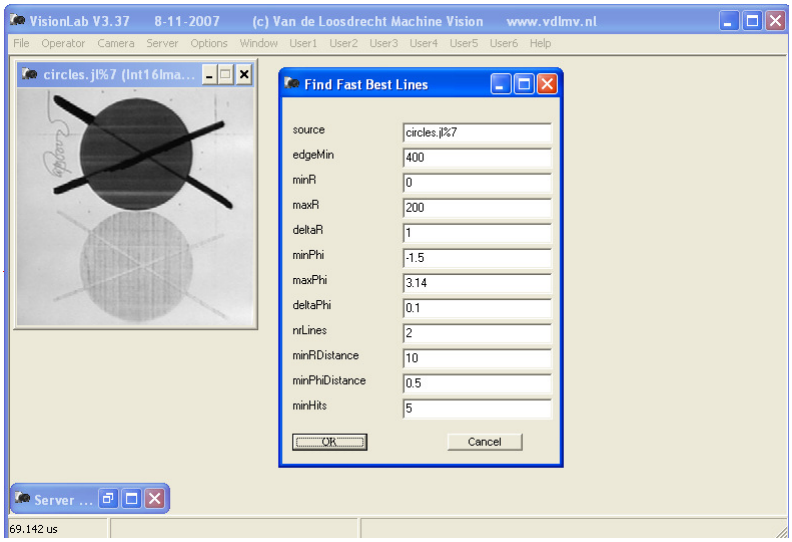
Find Best Lines (*)



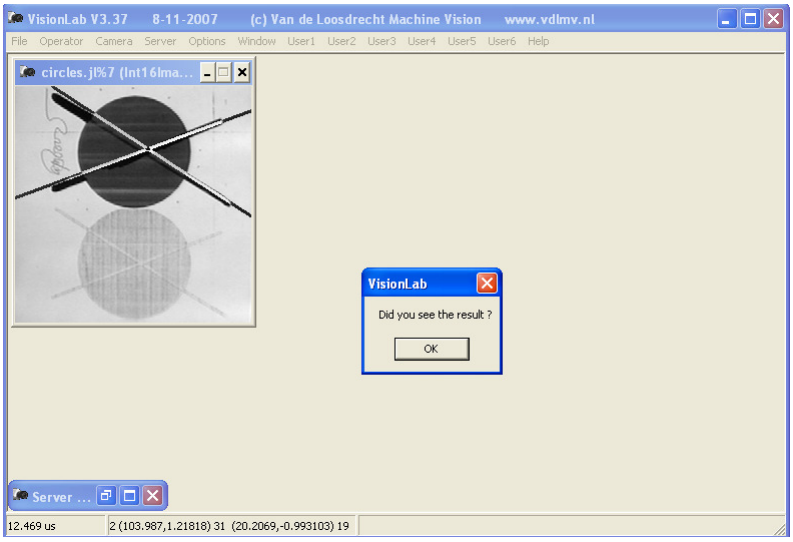
Demonstration Find Fast Best Lines

- **Open image circles.jl**
- **FindFastBestLines circles 7 400 0 200 1 -1.5 3.14 0.05 2 10 0.5 5**

Find Fast Best Lines



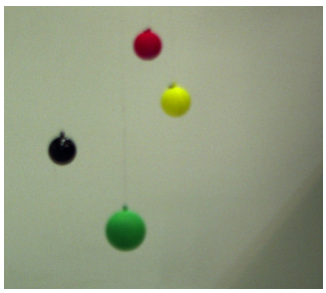
Find Fast Best Lines



27-aug-18 Distance and Hough Transforms 63

Exercise find balls

- **Use image robot_balls.jl**



- **Answer: ht_robot_balls.jls**

27-aug-18 Distance and Hough Transforms 64

Alternative for finding lines and circles

Alternative operators to find lines and circles are based on the Edge detection, see the chapter about Edge detection

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**