Computer Vision

Binary Morphology

10 April 2018

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

Binary Morphology

Overview:

- **Erosion**
- **Dilation**
- **Opening**
- **Closing**
- **Hit and miss (*)**
- **Thinning (*)**
- **Thickening (*)**
- **Skeleton**

Erosion

Erosion (src, dest, mask)

The erosion operator initialises a destination image by sliding a mask across a source image.

The mask may be visualised as a probe that slides across the src image, testing the spatial nature of the image at every point x. Where the mask translated to x can be contained in the original image (by placing the centre (= origin) of the mask at x), then x belongs to the destination image.

Usage:

- **to peel blobs**
- **to implement more complex operators**

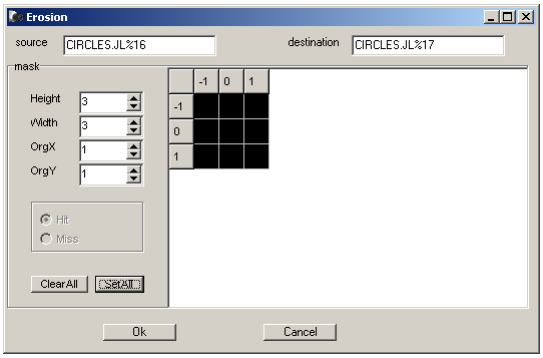
27-Aug-18 Binary Morphology 3

Demonstration Erosion

- **Open image circles.jl**
- **Threshold 0 130**
- **Erosion 3x3 setall**
- **Result: a layer of 1 pixel is eroded from image**
- **Eroded image (= 2nd) subtracted from thresholded image gives the border of objects**
- **Successive erosions erodes the legs and objects**

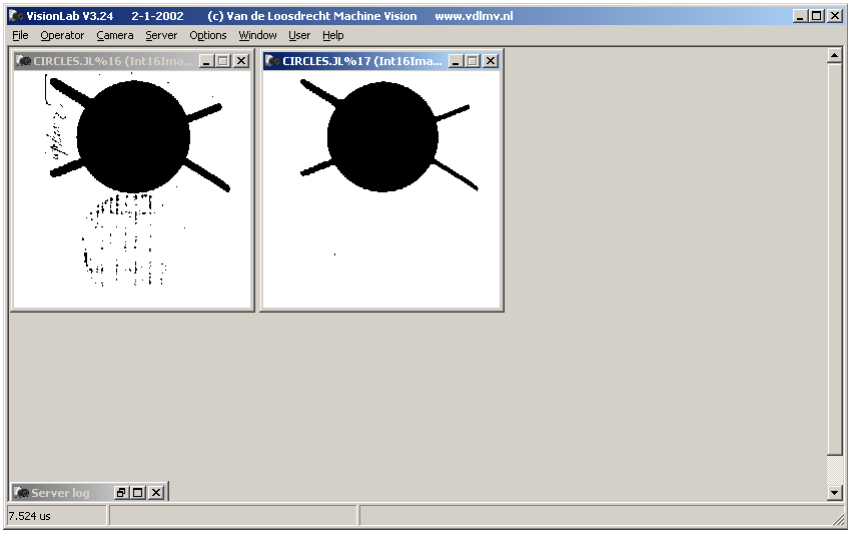
27-Aug-18 Binary Morphology 4

Erosion 3x3 setall



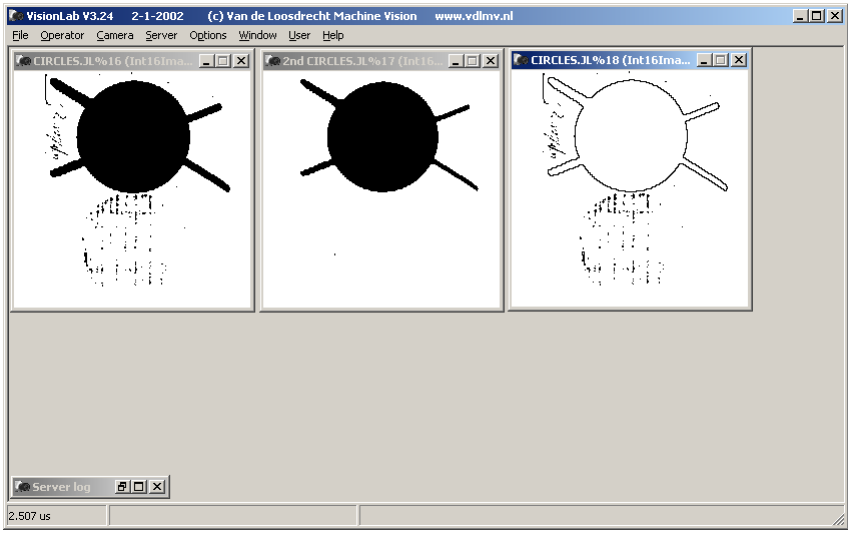
27-Aug-18 Binary Morphology 5

Erosion 3x3 setall



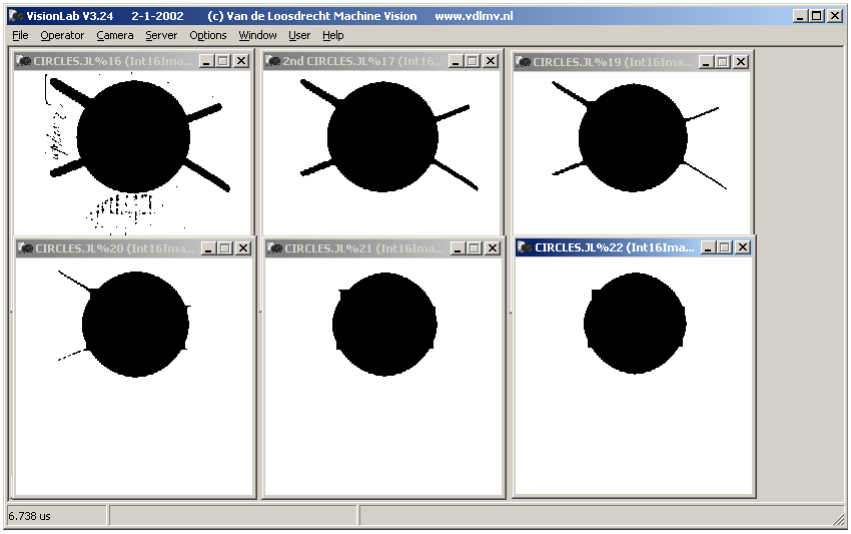
27-Aug-18 Binary Morphology 6

Eroded image (= 2nd) subtracted from Thresholded image gives the border of objects



27-Aug-18 Binary Morphology 7

Successive erosions erodes the legs and objects



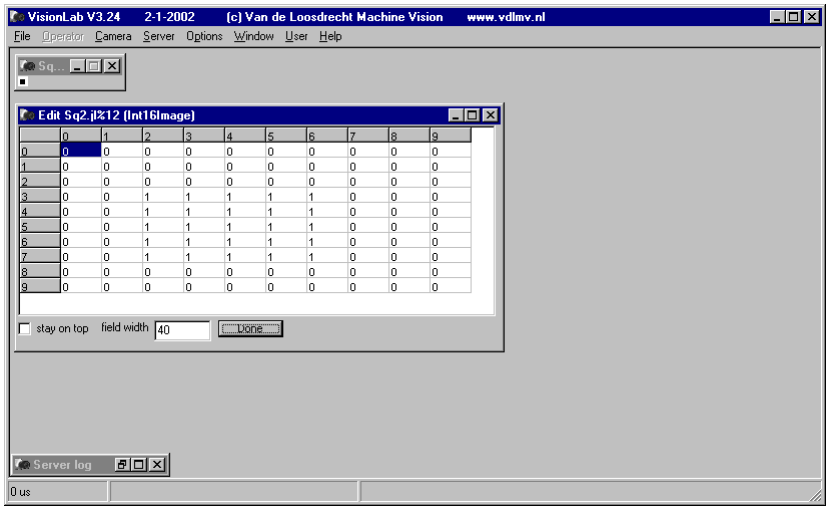
27-Aug-18 Binary Morphology 8

Demonstration Erosion

- **Open image sq2.jl**
- **Show image with edit**
- **Erosion with 1 1 1**
  0 0 1
  0 0 0
- **Show result with edit**
- **Explain why top, right and left border are removed**

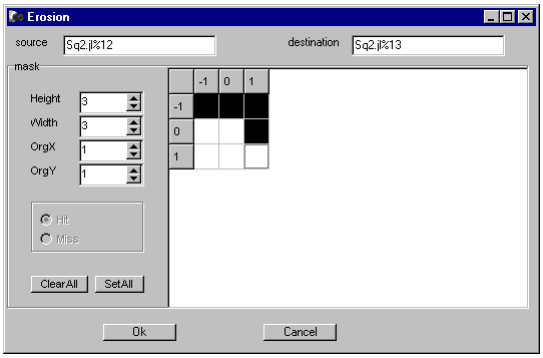
27-Aug-18 Binary Morphology 9

Image sq2.jl

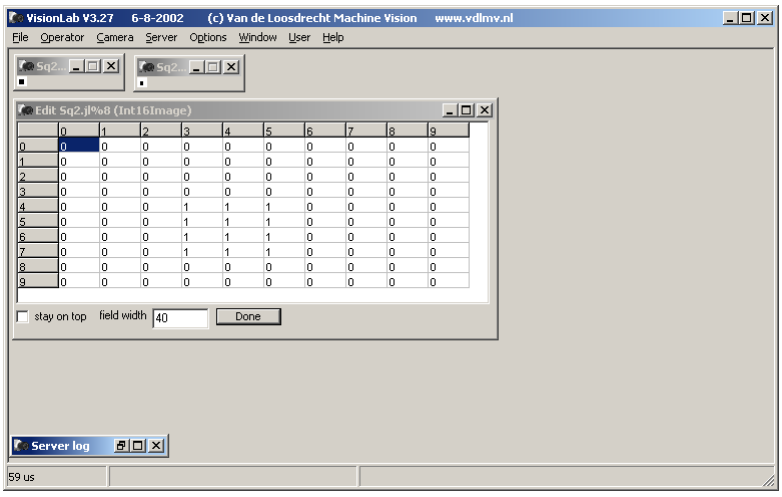


27-Aug-18 Binary Morphology 10

Erosion mask



Erosion



Dilation

Dilation (src, dest, mask)

At the start of the operation the destination image is filled with background (=0) pixel values. The mask is swept over the source image. Every time the centre (= origin) of the mask touches an object (=1) pixel in the source image, the entire translated mask is OR-ed to the destination image.

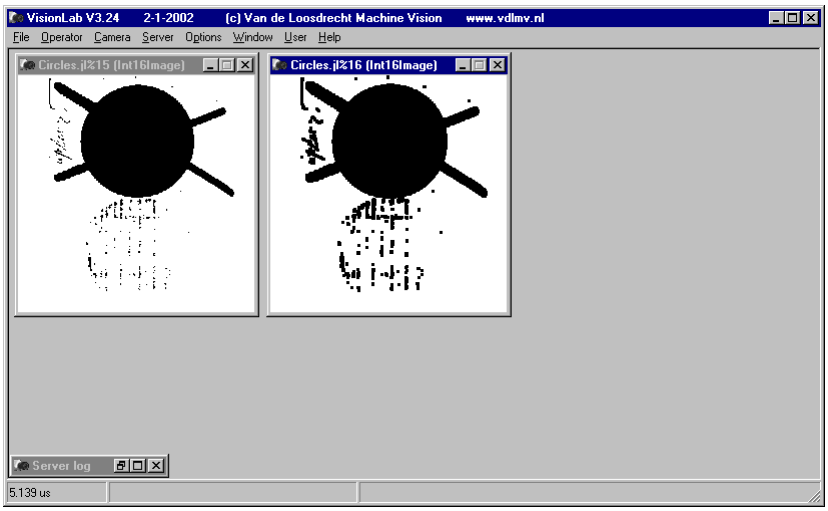
Usage:

- **to add a layer to blobs**
- **to implement more complex operators**

Demonstration Dilation

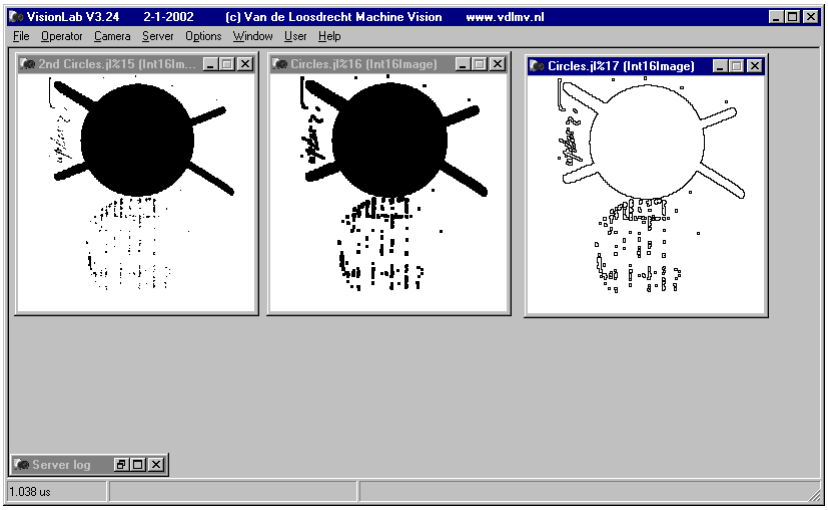
- **Open image circles.jl**
- **Threshold 0 130**
- **Dilation 3x3 setall**
- **Result: a layer of 1 pixel is added to the image**
- **Thresholded image (= 2nd) subtracted from Eroded image gives a border around the objects**
- **With successive dilations the objects grow**

Dilation 3x3 setall



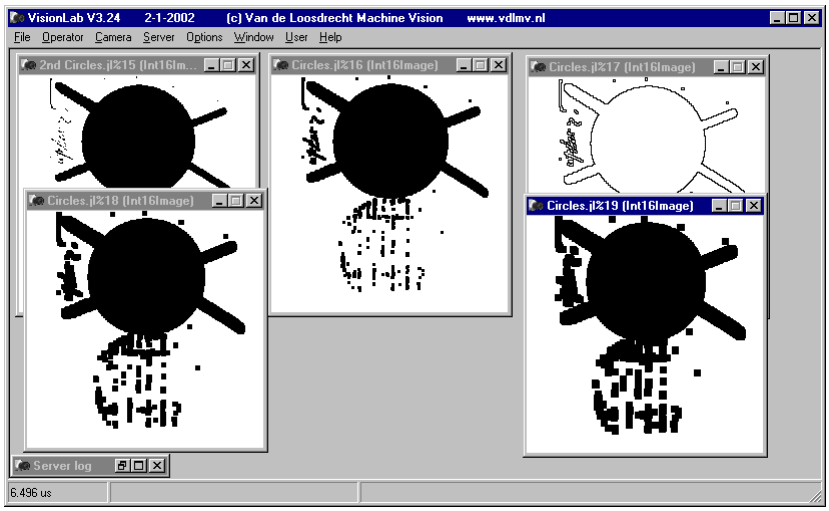
27-Aug-18 Binary Morphology 15

Thresholded image (= 2nd) subtracted from Eroded image gives a border around the objects



27-Aug-18 Binary Morphology 16

With successive dilations the objects grow



27-Aug-18 Binary Morphology 17

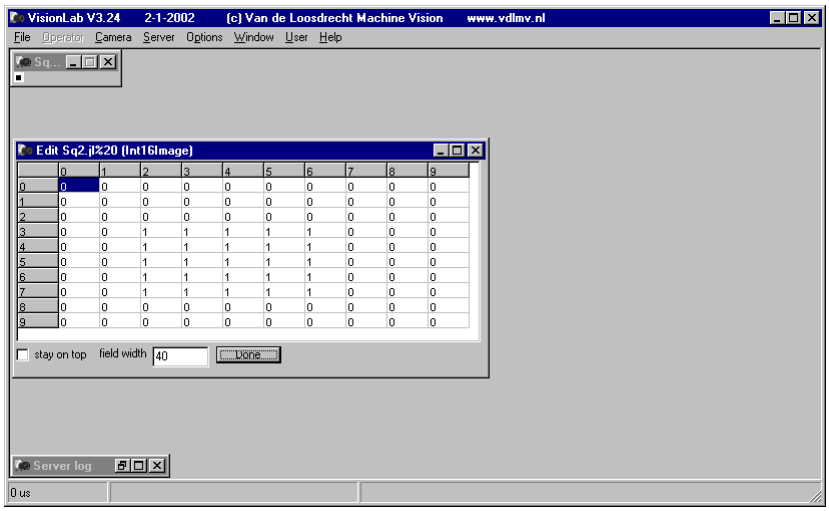
Demonstration Dilation

- **Open image sq2.jl**
- **Show image with edit**
- **Dilation with 0 0 0**
  0 0 0
  1 0 0
- **Show result with edit**
- **Explain why square is shifted one square to left bottom**

- **Same with Dilation with 0 0 0, result extra layer at left side**
  1 1 0
  0 0 0

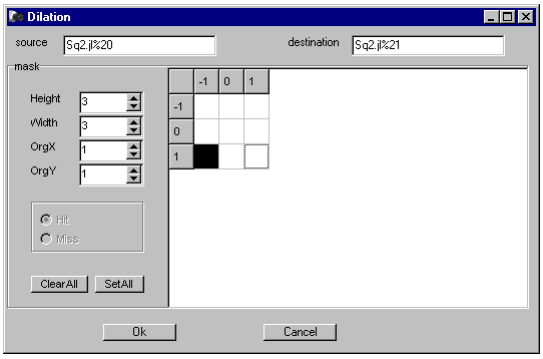
27-Aug-18 Binary Morphology 18

Image sq2.jl



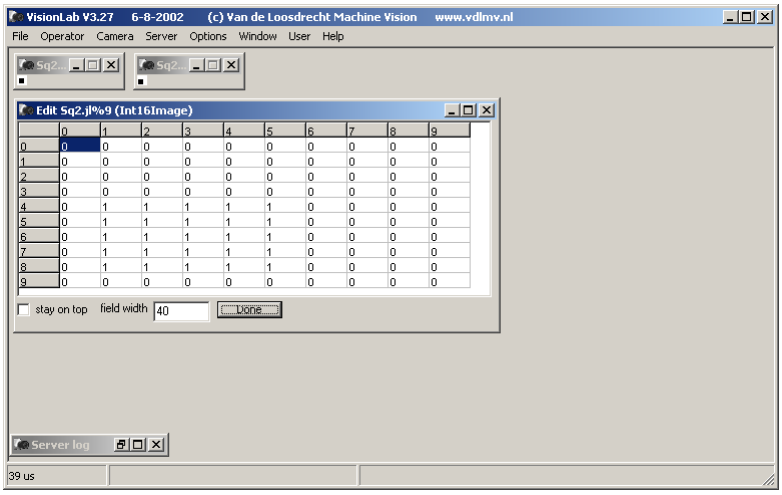
27-Aug-18 Binary Morphology 19

First example Dilation

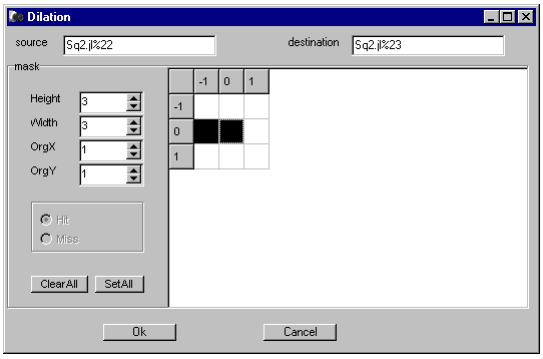


27-Aug-18 Binary Morphology 20

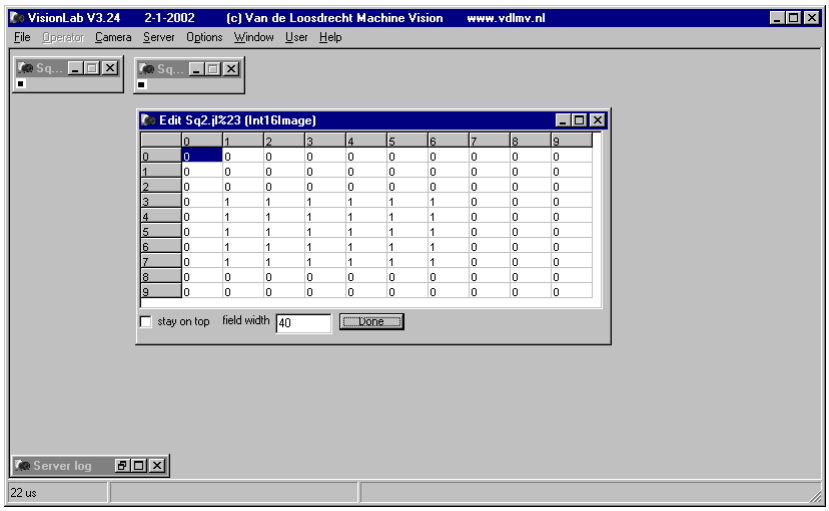
First example Dilation



Second example Dilation



Second example Dilation



Opening

Opening (src, dest, mask)

The opening operator initialises a destination image as the result of an erosion on the source image with the mask followed by a dilation with the same mask.

Usage: smooths the contour by

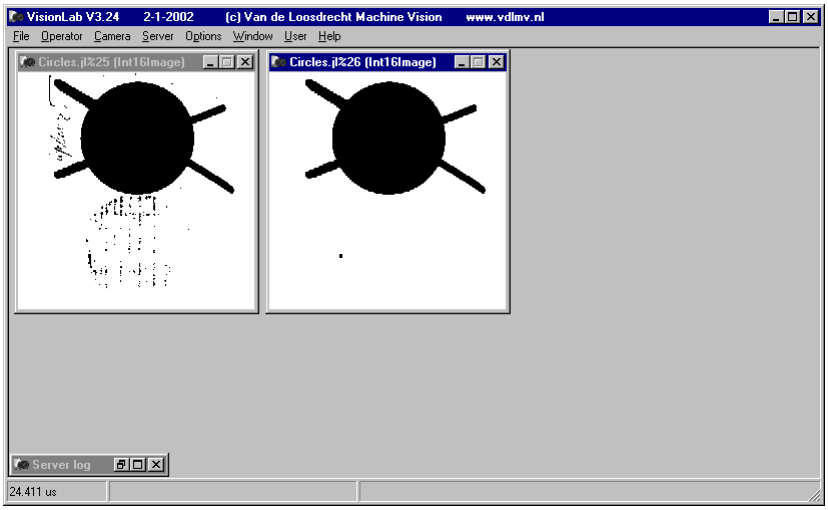
- **breaking narrow corridors**
- **eliminating small islands**
- **sharpening peaks**
- **small objects are removed and the original shape is almost retained**

Opening operation is idempotent

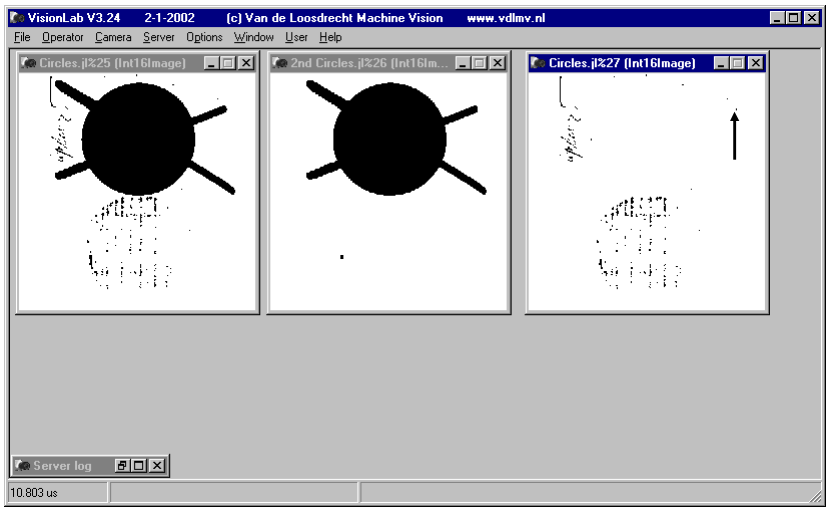
Demonstration Opening, removing small objects

- **Open image circles.jl**
- **Threshold 0 130**
- **Opening 3x3 setall**
- **Note small difference: difference between thresholded image and Opened image, right top leg, right edge (207,35), due to square shape erosion mask (*)**
- **2nd time same opening has no effect**

Opening 3x3 setall, removing small objects



Difference

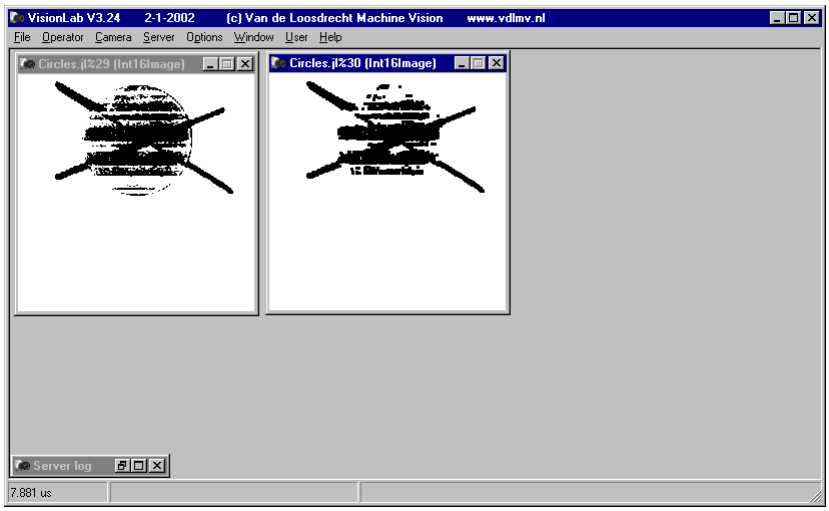


Demonstration Opening, sharpening of peaks

- **Open image circles.jl**
- **Threshold 0 60**
- **Opening 3x3 setall**
- **Show sharpening of peaks**

- **Implementation:**
  - **Erosion, followed by**
  - **Dilation with same mask**

Opening 3x3 setall, sharpening of peaks



27-Aug-18 Binary Morphology 29

Closing

Closing (src, dest, mask)

The closing operator initialises a destination image as the result of a dilation on the source image with the mask followed by an erosion with the same mask.

Usage: smooths the contour by

- **filling narrow breaks**
- **eliminating small holes**
- **filling gaps**

Closing operation is idempotent

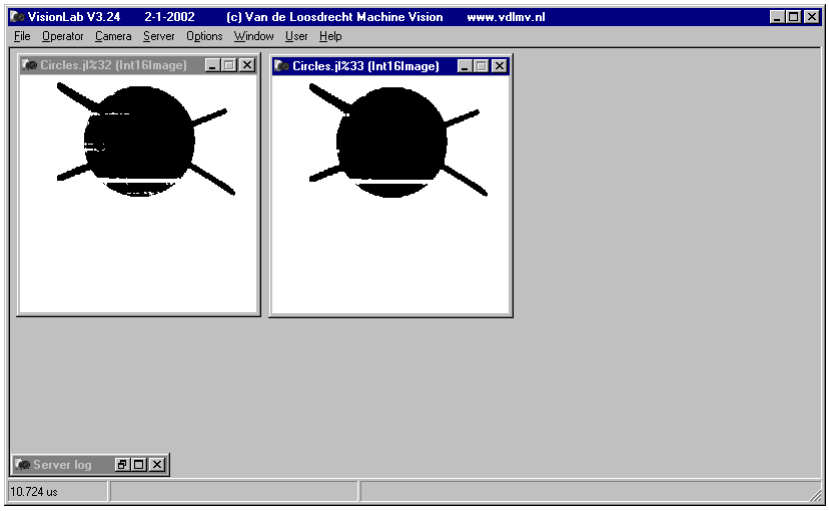
27-Aug-18 Binary Morphology 30

Demonstration Closing, filling gaps

- Open image circles.jl
- Threshold 0 70
- Closing 3x3 setall
- 2nd time same closing has no effect

- Implementation:
  - Dilation, followed by
  - Erosion with same mask

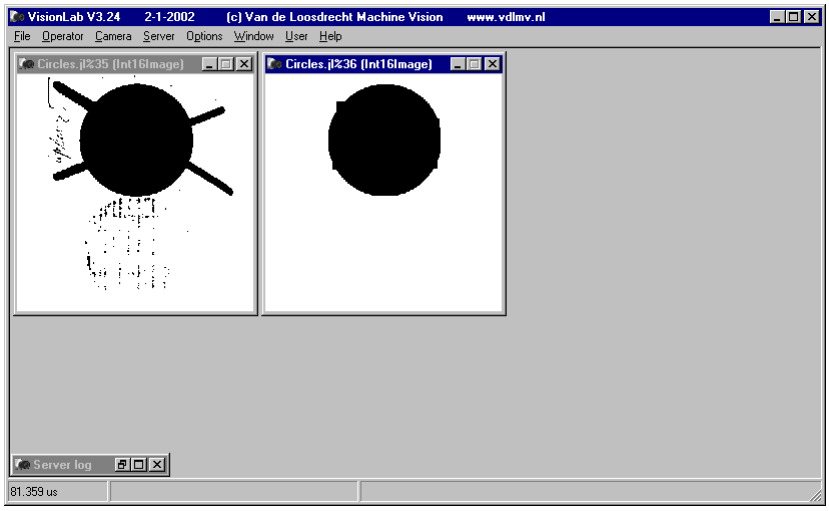
Closing 3x3 setall, filling gaps



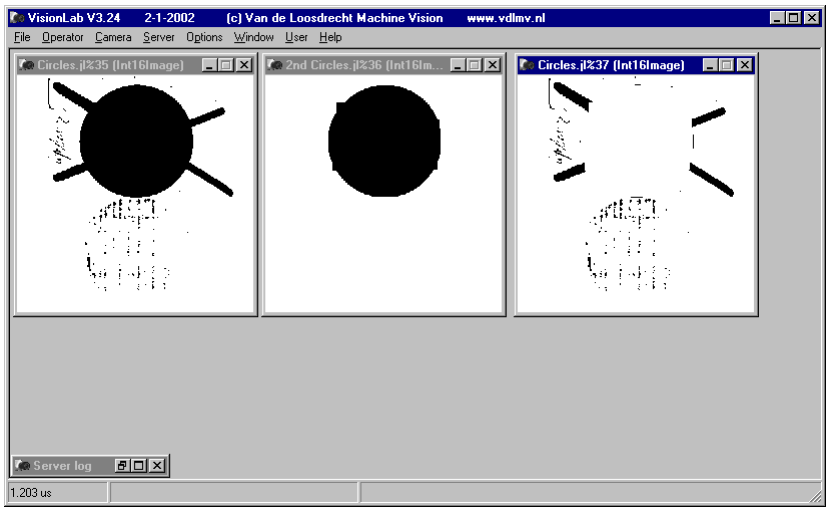
Demonstration Opening, find legs

- **Find the legs in threshold circles.jl 0 130:**
- **Opening 15x15 setall (org = 7,7), remove noise and legs**
- **Subtract Opened image (= 2nd) from thresholded image**
- **Opening 5x5 setall (org = 2,2) to remove noise**

Opening 15x15 setall (org = 7,7), remove noise and legs

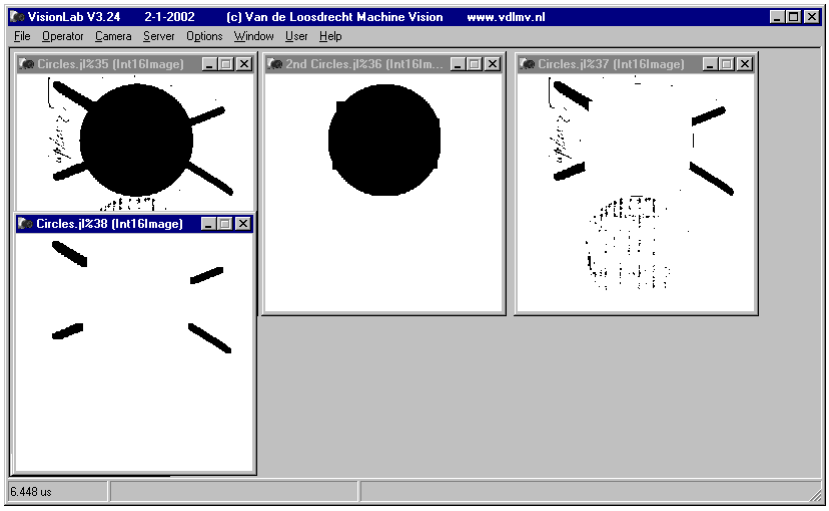


Subtract Opened image (= 2nd) from Thresholded image



27-Aug-18 Binary Morphology 35

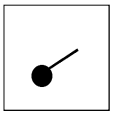
Opening 5x5 setall (org = 2,2) to remove noise



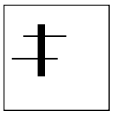
27-Aug-18 Binary Morphology 36

Exercises with Opening and Closing

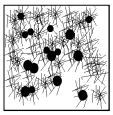
a) Use image ballbar.jl
Split ball from bar



b) Use image bars.jl
Split in one vertical bar and two horizontal bars



c) Use image ballstripes.jl
Filter the balls from the image



Answers:

- **ballbar.jls**
- **bars.jls**
- **ballstripes.jls**

27-Aug-18 Binary Morphology 37

Hit and Miss (*)

HitAndMiss (src, dest, hitMask, missMask)

The destination image has pixels set to the object value (= 1) at positions where all objects in the hitmask are contained in the source image and none of the object pixels in the missmask are contained in the source image.

Usage: to find specific points of blobs like corners or borders.

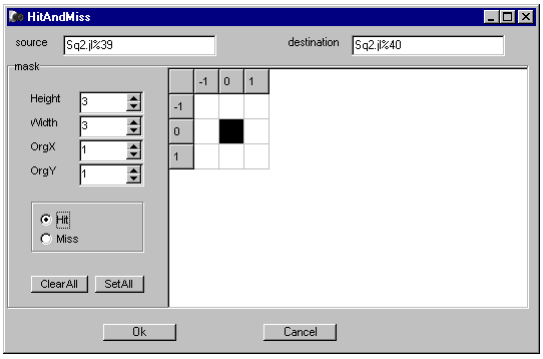
27-Aug-18 Binary Morphology 38

Demonstration Hit and Miss, find right border (*)

- **Example find right border**
  - **Open file sq2.jl**
  - **Apply HitAndMiss with hitmask: 0 0 0 and missmask: 0 0 1**
    0 1 0 0 0 1
    0 0 0 0 0 1

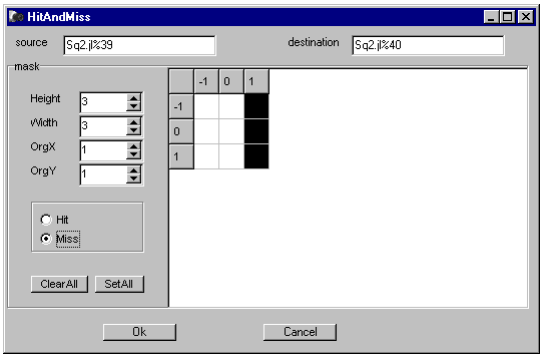
27-Aug-18 Binary Morphology 39

Hit mask (*)



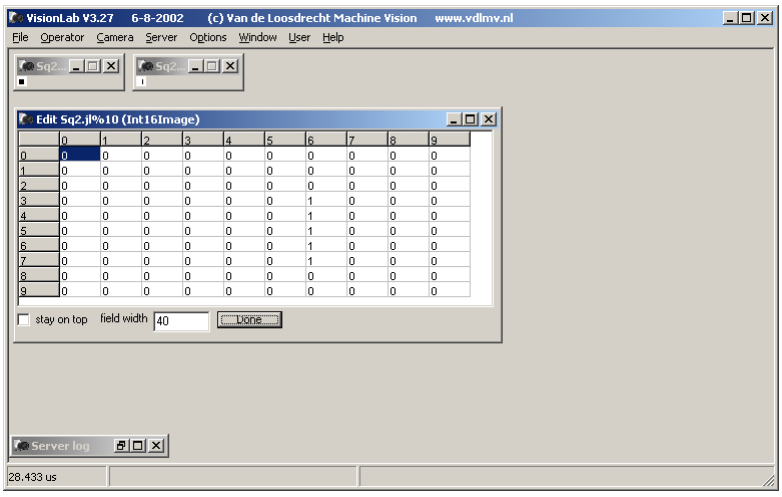
27-Aug-18 Binary Morphology 40

Miss mask (*)



27-Aug-18 Binary Morphology 41

Demonstration Hit and Miss, find right border (*)

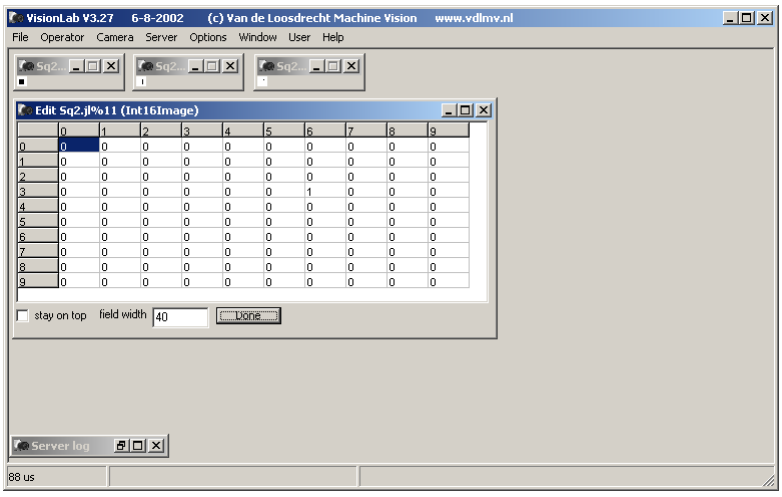


27-Aug-18 Binary Morphology 42

Demonstration Hit and Miss, find right top corner (*)

- **Example find right top corner**
  - **Open file sq2.jl**
  - **Apply HitAndMiss with hitmask: 0 0 0 and missmask: 0 1 1**
    1 1 0 0 0 1
    0 1 0 0 0 0

Demonstration Hit and Miss, find right top corner (*)



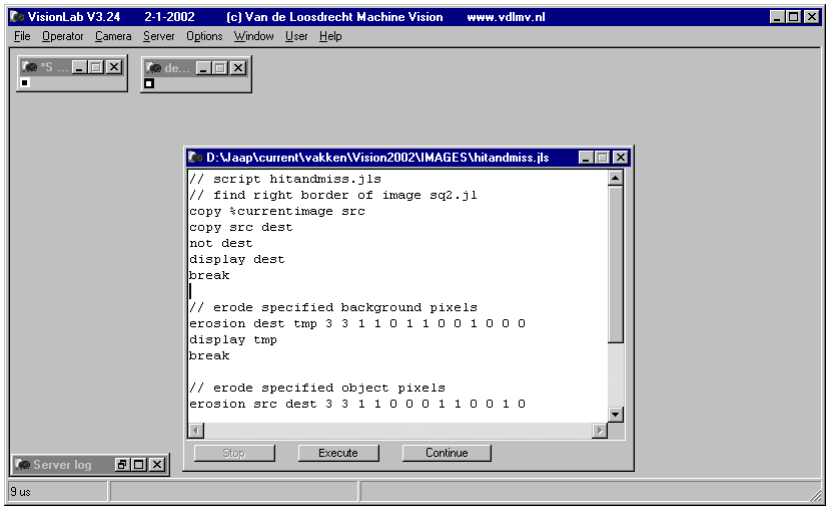
Hit and Miss (*)

Algorithm:

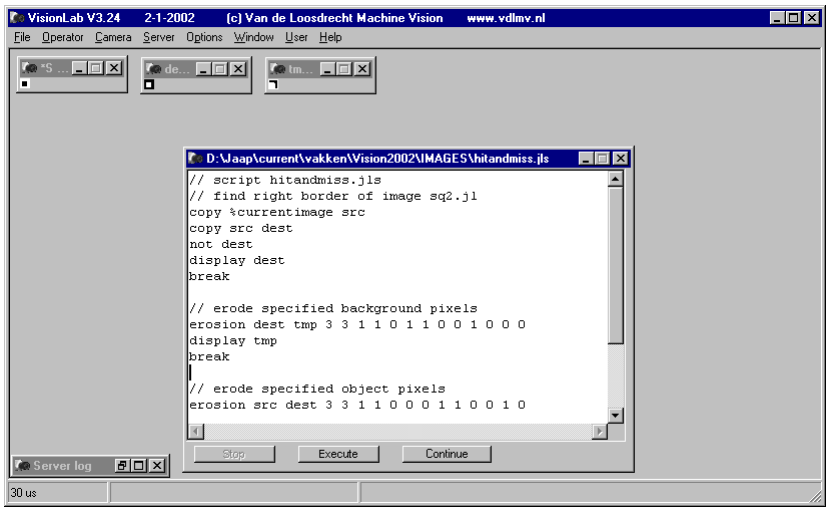
- **dest = src**
- **Not (dest)**
- **Erosion (dest, tmp, missMask) // 'dilate' background**
- **Erosion (src, dest, hitMask) // erode objects**
- **dest &= tmp;**

- **see script hitandmiss.jls**

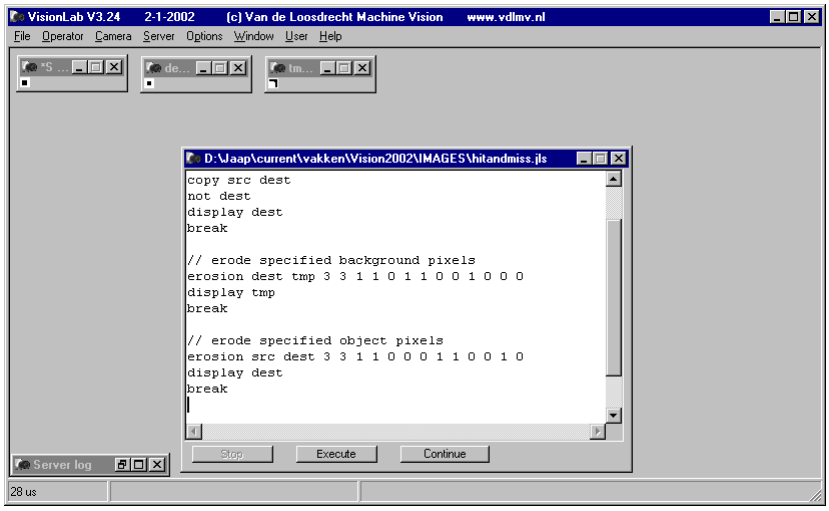
Not (dest) (*)



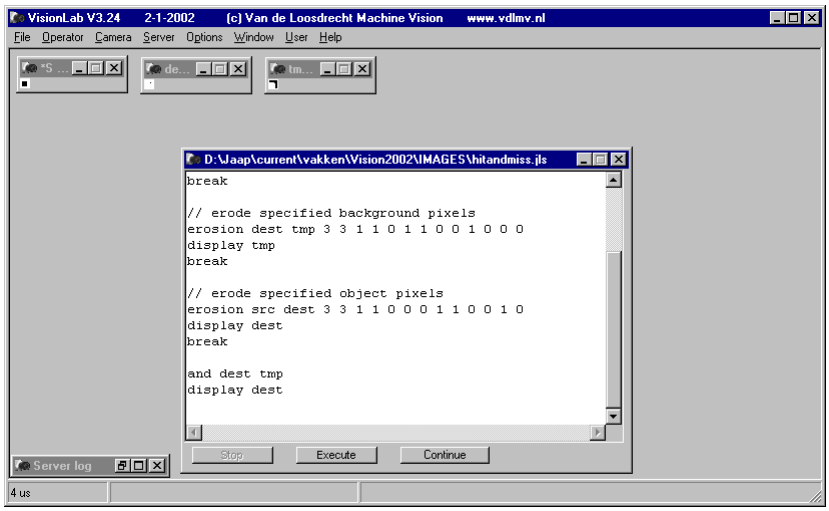
Erosion (dest, tmp, missMask) (*)



Erosion (src, dest, hitMask) (*)



dest &= tmp; (*)



Thinning (*)

Thinning (src, dest, hitMask, missMask)

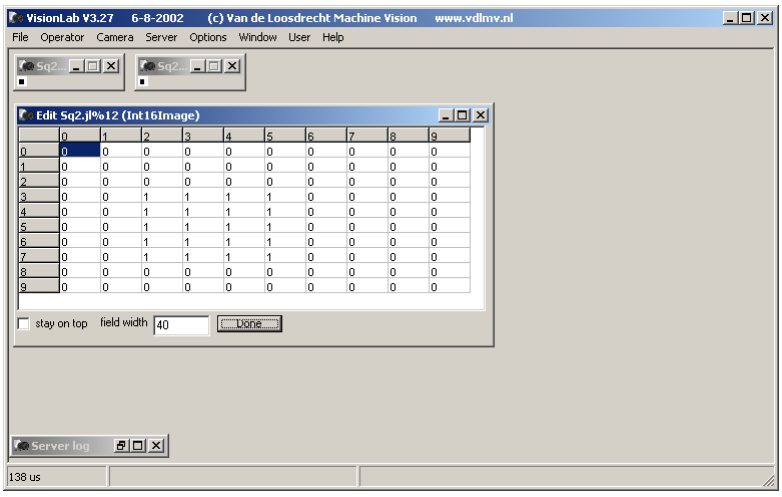
The destination image is the same as the source image but object pixels which are covered by the hitmask and not covered by the missmask are excluded from the destination image.
In order to perform meaningful, the origin of the hitmask should be *included* in the hitmask and the origin of the missmask should be *excluded* from the missmask.

Usage: at selective points peeling the blobs.

Demonstration Thinning, remove right border (*)

- **Open image sq2.jl**
- **Analyse image with Edit**
- **Apply Thinning with hitmask: 0 0 0 and missmask: 0 0 1**
  0 1 0 0 0 1
  0 0 0 0 0 1
- **Analyse result with Edit; right border has been removed**

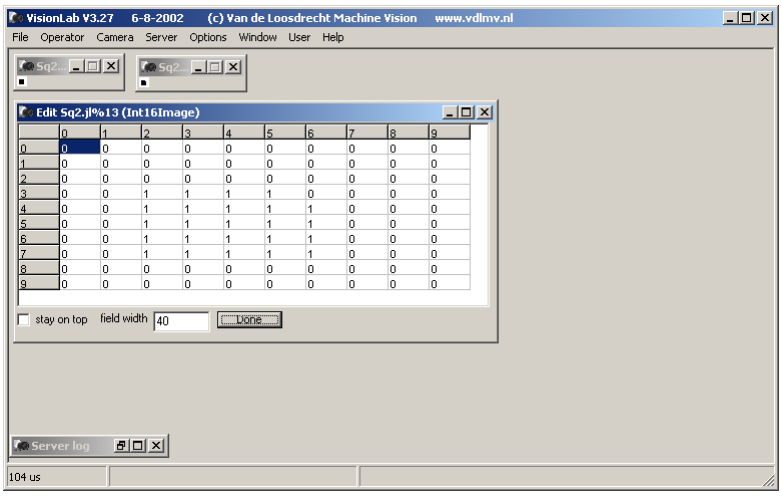
Demonstration Thinning, remove right border (*)



Demonstration Thinning, remove right top (*)

- **Open image sq2.jl**
- **Analyse image with Edit**
- **Apply Thinning with hitmask: 0 0 0 and missmask: 0 1 1**
  1 1 0 0 0 1
  0 1 0 0 0 0
- **Analyse result with Edit; right top has been removed**

Demonstration Thinning, remove right top (*)



Thinning (*)

Algorithm:

- **HitAndMiss (src, tmp, hitMask, missMask)**
- **dest = src**
- **dest -= tmp (exor is also possible)**

Thickening (*)

Thickening (src, dest, hitMask, missMask)

The destination image is the same as the source image but at special positions pixels are added.

These positions are where all objects in the hitmask are contained in the source image and none of the object pixels in the missmask are contained in the source image.

In order to perform meaningful, the origin of the hitmask should be *excluded* from the hitmask and the origin of the missmask should be *included* in the missmask.
Note: only the origin pixel is added.

Usage: add at selective points a layer to the blobs

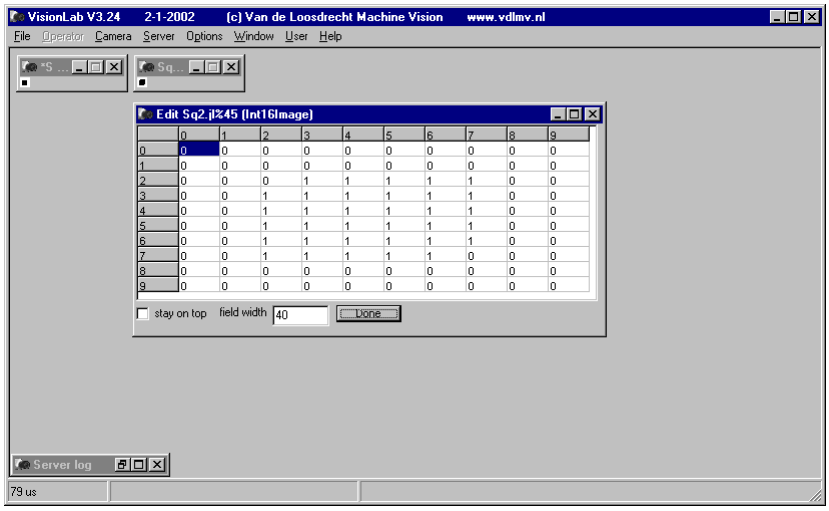
Demonstration Thickening (*)

- **Open image sq2.jl**
- **Analyse image with Edit**
- **Apply Thickening with hitmask: 0 0 0 and missmask: 0 0 0**
  0 0 0 0 1 0
  1 0 0 0 0 0

  must be present

  is not present and is added
- **Analyse result with Edit; a layer is added at the top and the right side, note top left and bottom right.**

Demonstration Thickening (*)



Thickening (*)

Algorithm:

- **HitAndMiss (src, dest, hitMask, missMask)**
- **dest |= src (adding also possible)**

Skeleton

Skeleton (sourceImage, destinationImage)

Function:

- **Calculates the skeleton of all blobs in a binary image**

Algorithm:

- **Iterative Thinning (= "grass burning")**

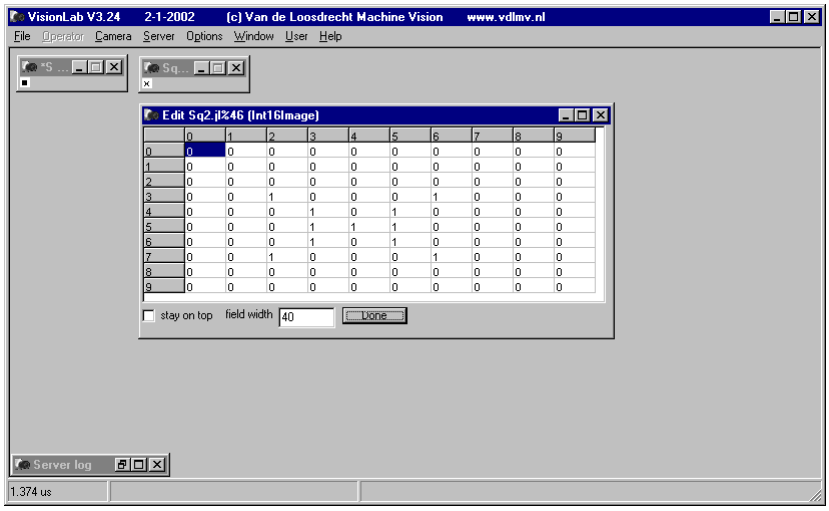
Usage:

- **To get the 'basic form' of a blob**
- **Slow operation**

Demonstration Skeleton sq.jl

- **Open image sq.jl**
- **Apply Skeleton**
- **Analyse result with Edit**

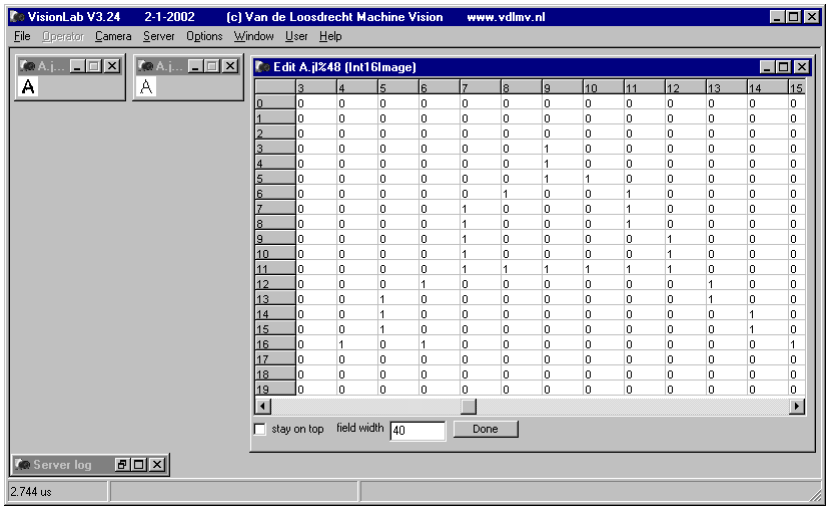
Demonstration Skeleton sq.jl



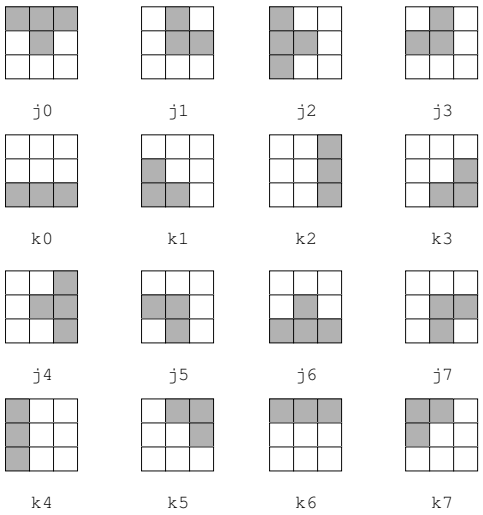
Demonstration Skeleton A.jl

- **Open image A.jl**
- **Apply Skeleton**
- **Analyse result with Edit**

Demonstration Skeleton A.jl



Skeleton implementation masks (*)



Skeleton algorithm (*)

```
dest = src;
do {
  prevImage = dest;
  for (int i = 0; i < 8; i += 2) {
            Thinning (dest, tmp, j[i], k[i]);
            Thinning (tmp, dest, j[i+1], k[i+1]);
  }
} while (prevImage != dest);
```

Note:

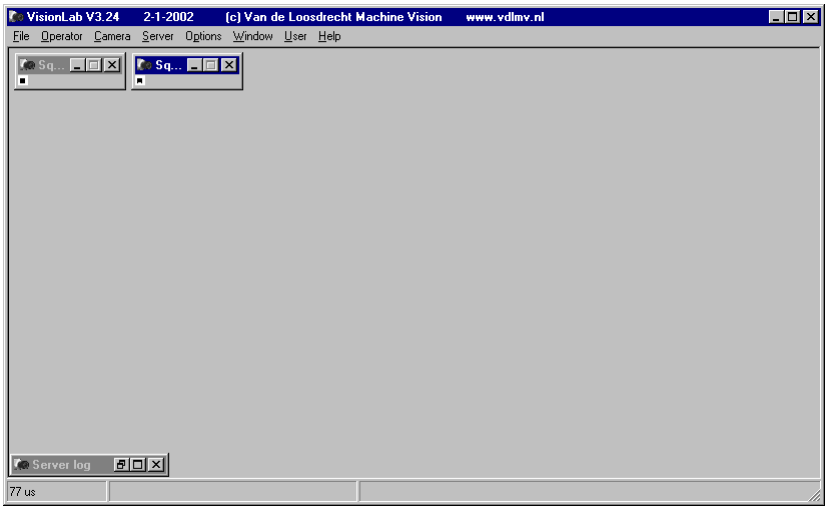
- **This is a very slow implementation**
- **Alternatives can be based on distance transform**

Demonstration Skeleton algorithm (*)

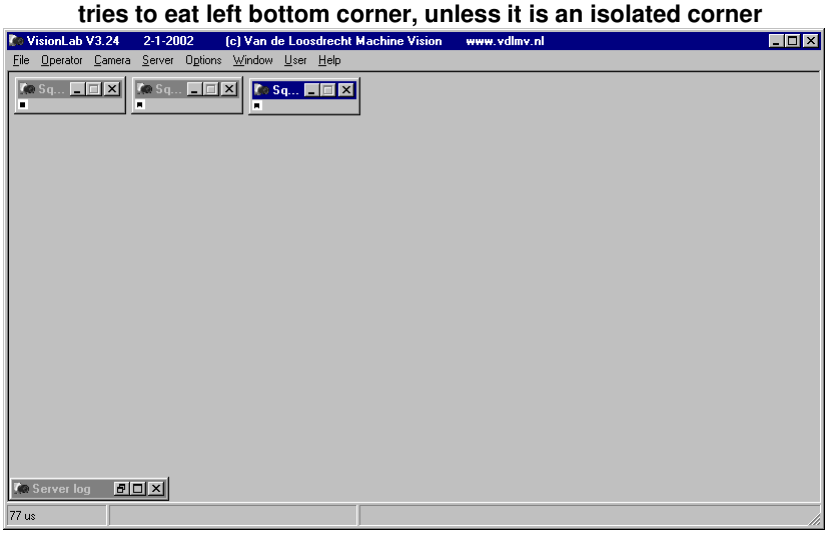
- **Open image sq2.jl**
- **Thinning with masks j0 and k0,**
  eats middle of bottom border, but excludes the corners
- **Thinning result with masks j1 and k1,**
  tries to eat left bottom corner, unless it is an isolated corner

- **To see that Thinning with masks j1 and k1,**
  tries to eat left bottom corner, unless it is an isolated corner,
  do this operation on sq2.jl

Thinning with masks j0 and k0, (*)
eats middle of bottom border, but excludes the corners

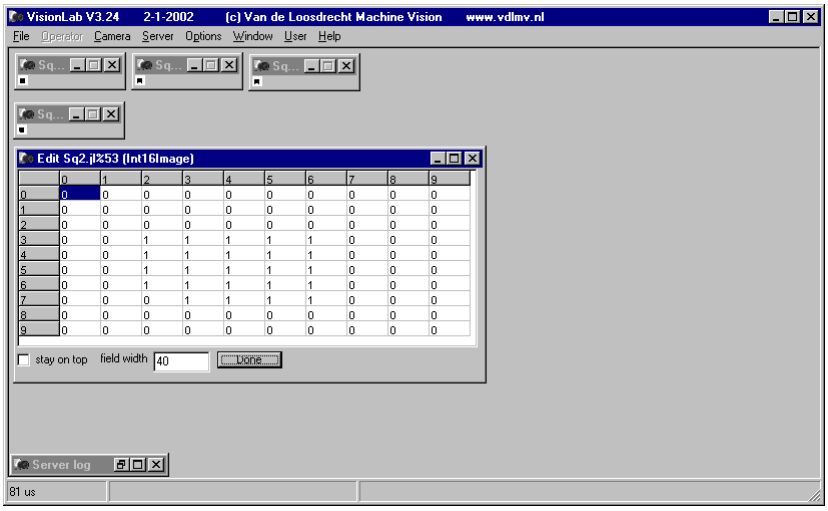


Thinning result with masks j1 and k1, (*)
tries to eat left bottom corner, unless it is an isolated corner



27-Aug-18 Binary Morphology 69

Thinning with masks j1 and k1 on sq.jl, (*)
tries to eat left bottom corner, unless it is an isolated corner



27-Aug-18 Binary Morphology 70

End points and branch points

In skeletons special points can be identified:

- **end points (= one object neighbour)**
- **branch points (= more then two object neighbours)**

NrOfNeighbours srcImage destImage connected

All background pixels will get the value 0.

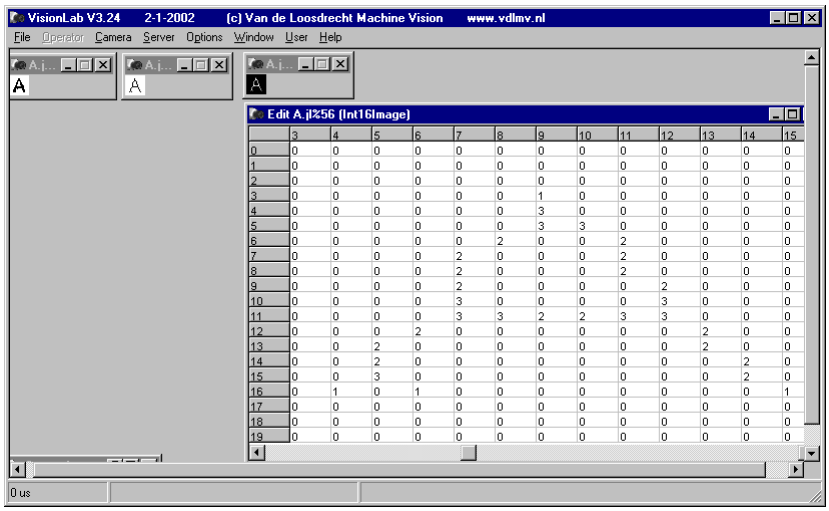
All pixels belonging to a blob will get the value of its number of neighbours.

The parameter connected has the value EightConnected or FourConnected and determines how the blobs are connected.

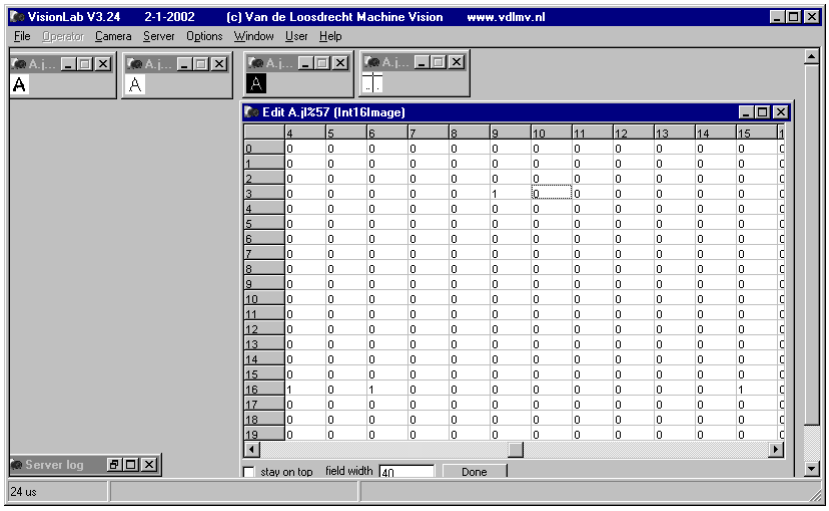
Demonstration finding endpoints

- **Open image A.jl**
- **Skeleton**
- **NrOfNeighbours EightConnected (from the segmentation menu)**
- **Show results with edit**
- **Threshold 1 1,**
  to find endpoints, note there are 4 !

NrOfNeighbours



Threshold 1 1, to find 4 endpoints



Implementation of NrOfNeigbours (*)

For EightConnected objects:

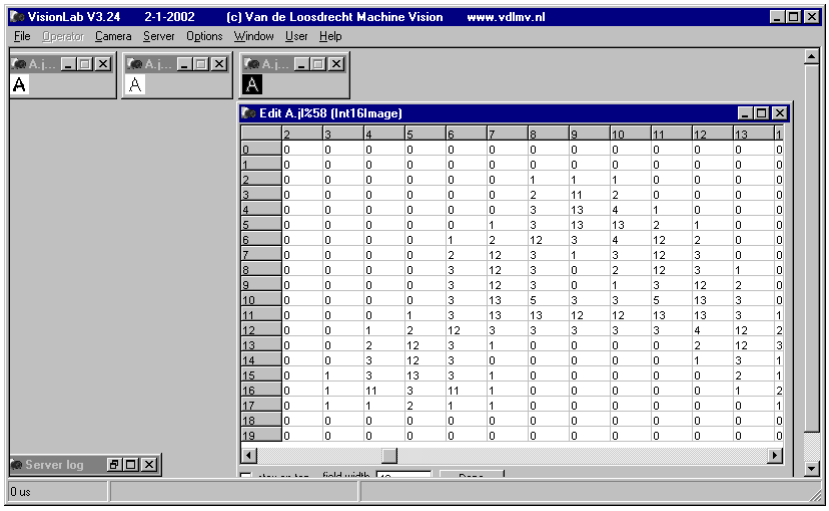
- **Convolution with mask**

  1 1 1
  1 10 1
  1 1 1

  - **p < 10: background pixel with p number of objects as neighbour**
  - **p > 10: object pixel with p - 10 number of objects as neighbour**
- **Subtract 10 from all pixels**
- **Setselectedtovalue -10 -1 0**

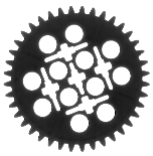
27-Aug-18 Binary Morphology 75

Convolution on skeleton (*)



27-Aug-18 Binary Morphology 76

Exercise count number of tooth on gear



- **Use image gear.jl**
- **Write script for counting number of teeth**

- **answer script gear.jls**