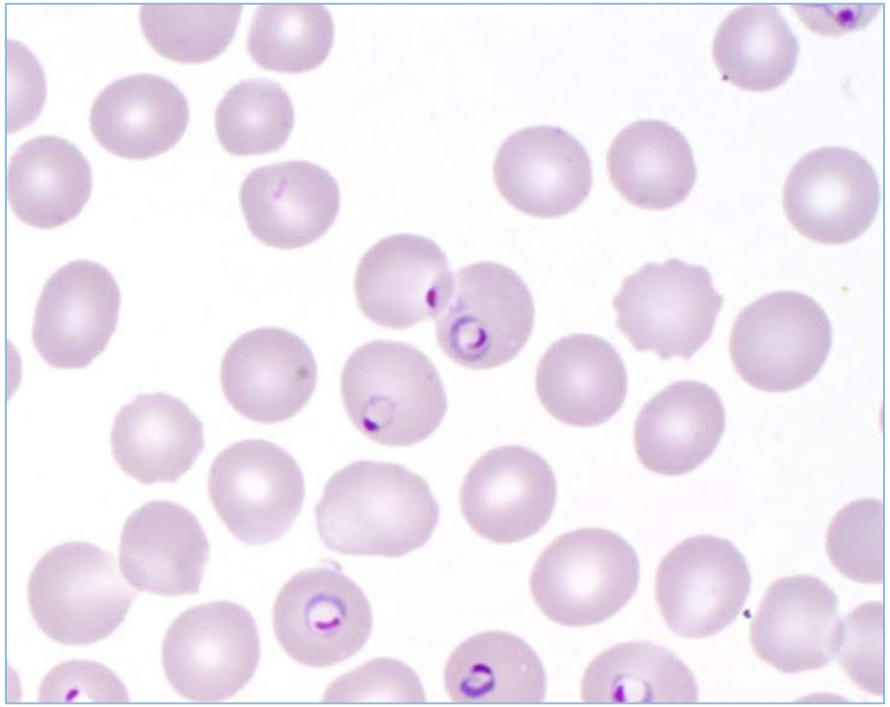
Case study #1: Object counting

Lecturer: Bui Ha Duc, PhD

Email: ducbh@hcmute.edu.vn

Target

- Count the number of target objects in a image



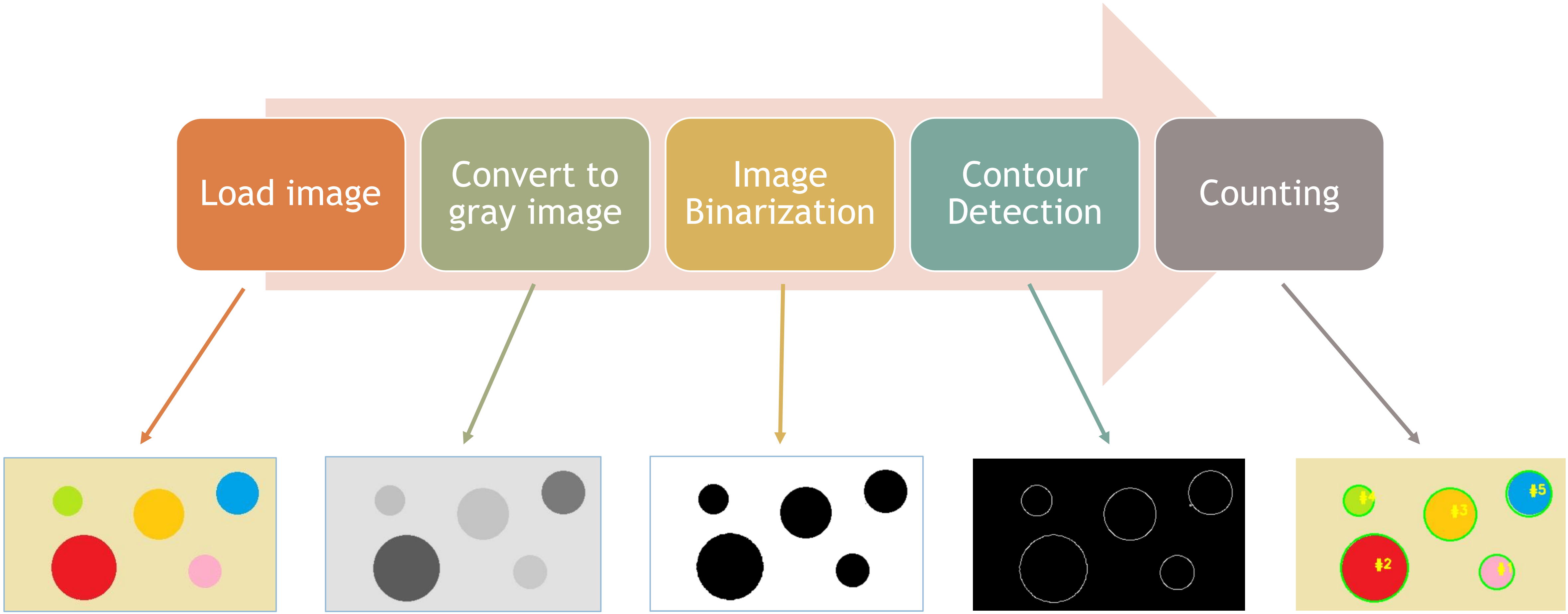
How many cells in this image?

Knowledge will be covered

In this case you will learn

- How to convert an image to gray
- How to binarize a gray image using threshold techniques
- How to find object contours
- How to mark and label object on an image

Simple Object Counting Procedure



Color Conversion

- RGB to Gray

$$\text{RGB[A] to Gray:} \quad Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

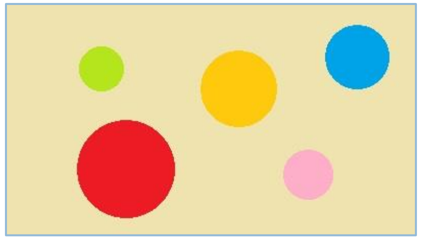
- OpenCV Function:

```
cvtColor(src, gray, COLOR_BGR2GRAY);
```

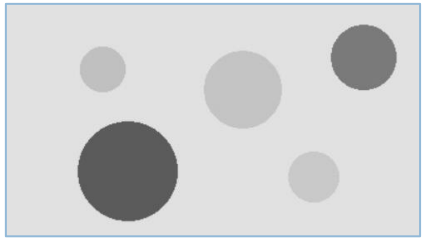
Conversion mode

- COLOR_BGR2GRAY
- COLOR_BGR2HSV
- COLOR_HSV2BGR
- COLOR_BGR2Lab

Source Image



Gray Image



```
// load image file
Mat img = imread(argv[1],1);
// convert image to gray
Mat gray;
cvtColor(img, gray,COLOR_BGR2GRAY);
```

https://docs.opencv.org/4.1.1/de/d25/imgproc_color_conversions.html

Thresholding

- Purpose: convert a gray image to a binary image
- OpenCV function:

```
threshold(src,dst,threshold_value,max_value,threshold_type);
```

```
threshold_value (THRESH_BINARY):
if pixel_value < threshold_value
    pixel_value = 0;
else
    pixel_value = max value;
```

```
// convert to binary image using threshold
Mat b_img;
double thresh = 115;
double maxValue = 255;

threshold(gray,b_img,thresh,maxValue,THRESH_BINARY);
```

https://docs.opencv.org/4.1.1/db/d8e/tutorial_threshold.html

Thresholding

- Threshold Types

Enumerator	
THRESH_BINARY Python: cv.THRESH_BINARY	$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_BINARY_INV Python: cv.THRESH_BINARY_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$
THRESH_TRUNC Python: cv.THRESH_TRUNC	$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
THRESH_TOZERO Python: cv.THRESH_TOZERO	$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_TOZERO_INV Python: cv.THRESH_TOZERO_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
THRESH_MASK Python: cv.THRESH_MASK	
THRESH_OTSU Python: cv.THRESH_OTSU	flag, use Otsu algorithm to choose the optimal threshold value
THRESH_TRIANGLE Python: cv.THRESH_TRIANGLE	flag, use Triangle algorithm to choose the optimal threshold value

https://docs.opencv.org/4.1.1/d7/d1b/group__imgproc__misc.html#gaa9e58d2860d4afa658ef70a9b1115576

Contour Detection

- **Contours**
  - curve joining all the continuous points (along the boundary), having same color or intensity.
  - are a useful tool for shape analysis, object detection and recognition.
- OpenCV Function:

`findContours(src,contours,hierarchy,find_mode,approx_method,offset)`

```
vector<vector<Point> > contours;
findContours( img, contours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE );
```

contours
Detected contours. Each contour is stored as a vector of vectors of points.

hierarchy (Optional)
Output vector, containing information about the image topology.

find_mode
Contour retrieval mode

approx_method
Contour approximation method

https://docs.opencv.org/4.1.1/d3/dc0/group__imgproc__shape.html#gadf1ad6a0b82947fa1fe3c3d497f260e0

Contour Detection

- **Contour Retrieval Mode**

Enumerator	
RETR_EXTERNAL Python: cv.RETR_EXTERNAL	retrieves only the extreme outer contours. It sets `hierarchy[i][2]=hierarchy[i][3]=-1` for all the contours.
RETR_LIST Python: cv.RETR_LIST	retrieves all of the contours without establishing any hierarchical relationships.
RETR_CCOMP Python: cv.RETR_CCOMP	retrieves all of the contours and organizes them into a two-level hierarchy. At the top level, there are external boundaries of the components. At the second level, there are boundaries of the holes. If there is another contour inside a hole of a connected component, it is still put at the top level.
RETR_TREE Python: cv.RETR_TREE	retrieves all of the contours and reconstructs a full hierarchy of nested contours.
RETR_FLOODFILL Python: cv.RETR_FLOODFILL	

https://docs.opencv.org/4.1.1/d9/d8b/tutorial_py_contours_hierarchy.html

Contour Detection

- Approximation Modes

Enumerator	
CHAIN_APPROX_NONE Python: cv.CHAIN_APPROX_NONE	stores absolutely all the contour points. That is, any 2 subsequent points (x1,y1) and (x2,y2) of the contour will be either horizontal, vertical or diagonal neighbors, that is, max(abs(x1-x2),abs(y2-y1))==1.
CHAIN_APPROX_SIMPLE Python: cv.CHAIN_APPROX_SIMPLE	compresses horizontal, vertical, and diagonal segments and leaves only their end points. For example, an up-right rectangular contour is encoded with 4 points.
CHAIN_APPROX_TC89_L1 Python: cv.CHAIN_APPROX_TC89_L1	applies one of the flavors of the Teh-Chin chain approximation algorithm [212]
CHAIN_APPROX_TC89_KCOS Python: cv.CHAIN_APPROX_TC89_KCOS	applies one of the flavors of the Teh-Chin chain approximation algorithm [212]

Prolem 1: Image noise

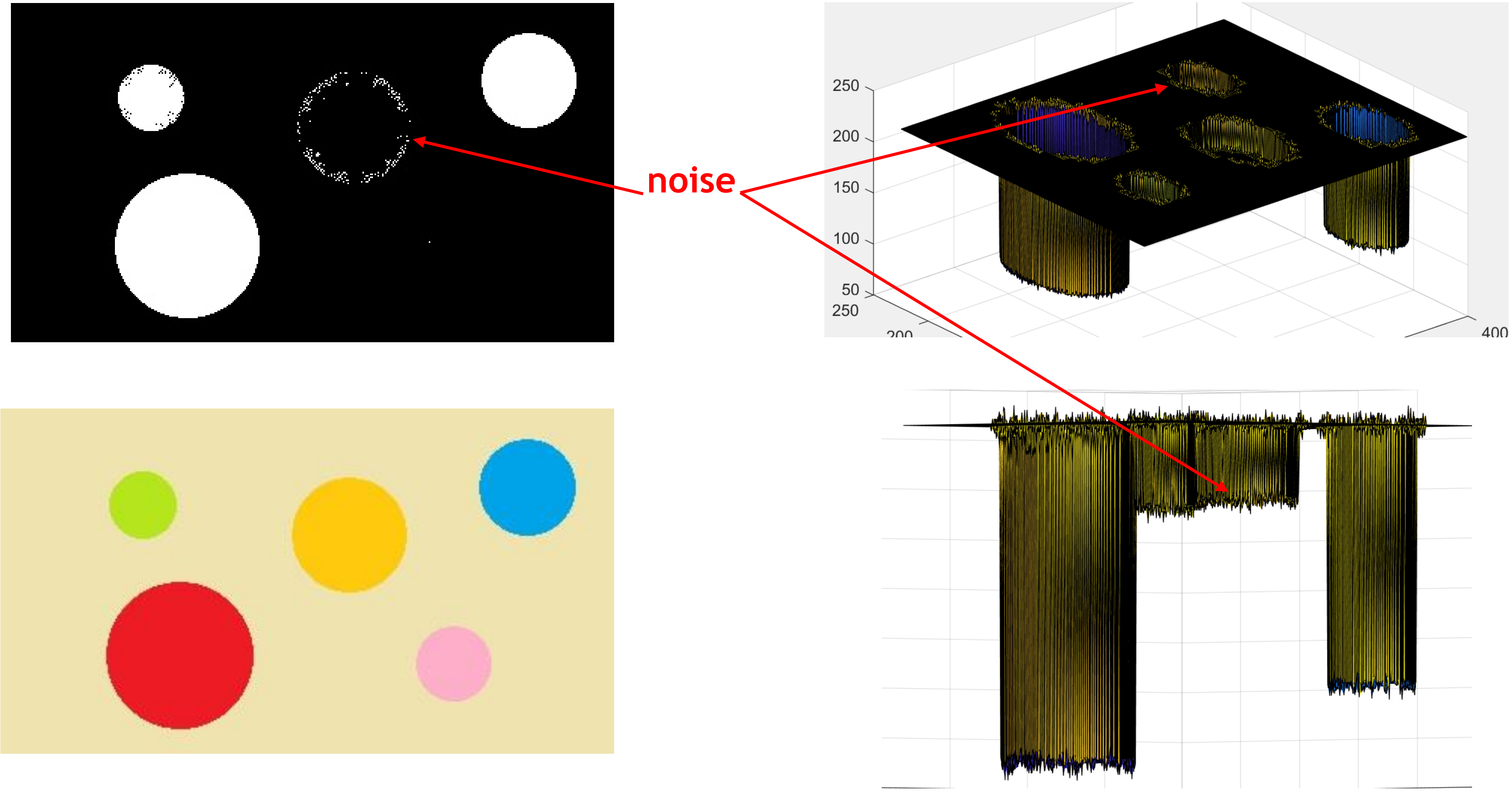
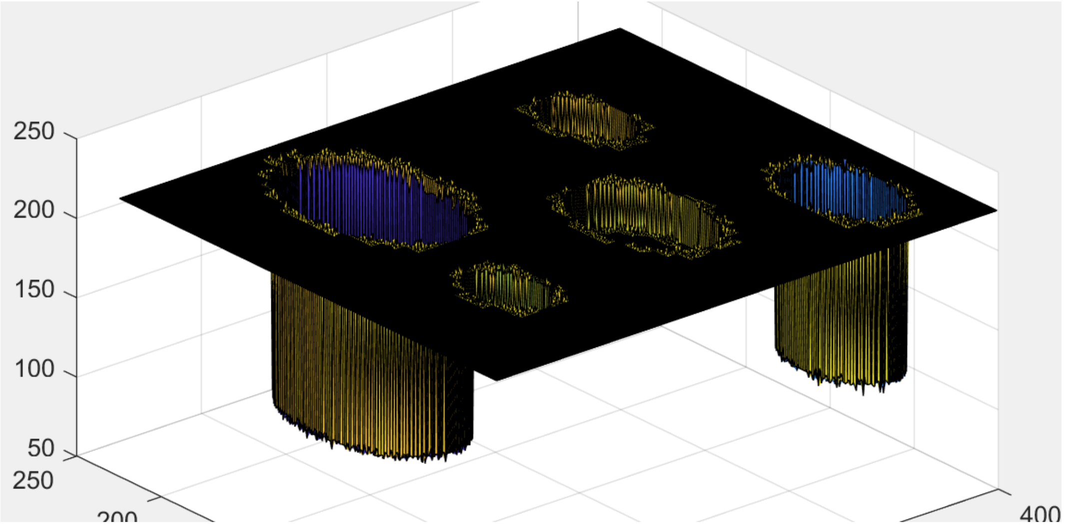


Image smoothing

- Image smoothing: is a key technique in image processing used to reduces noise within an image



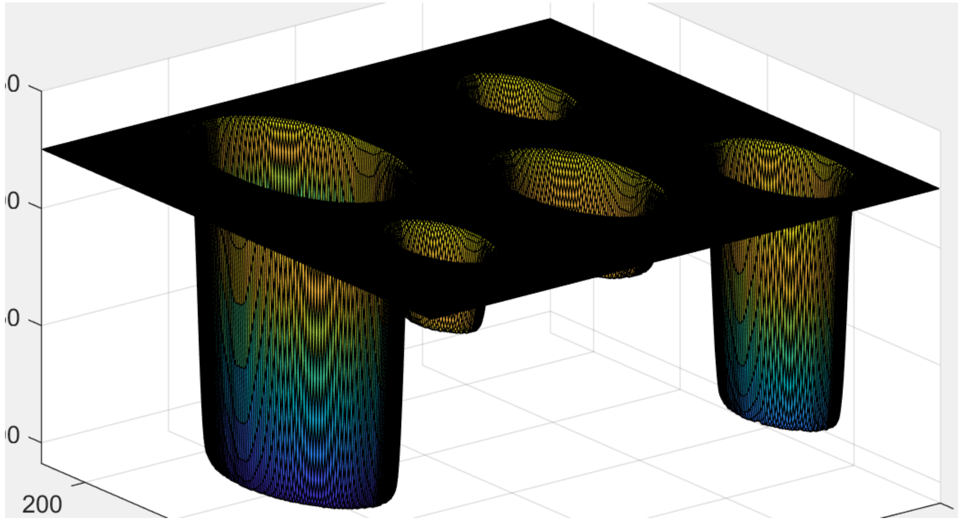
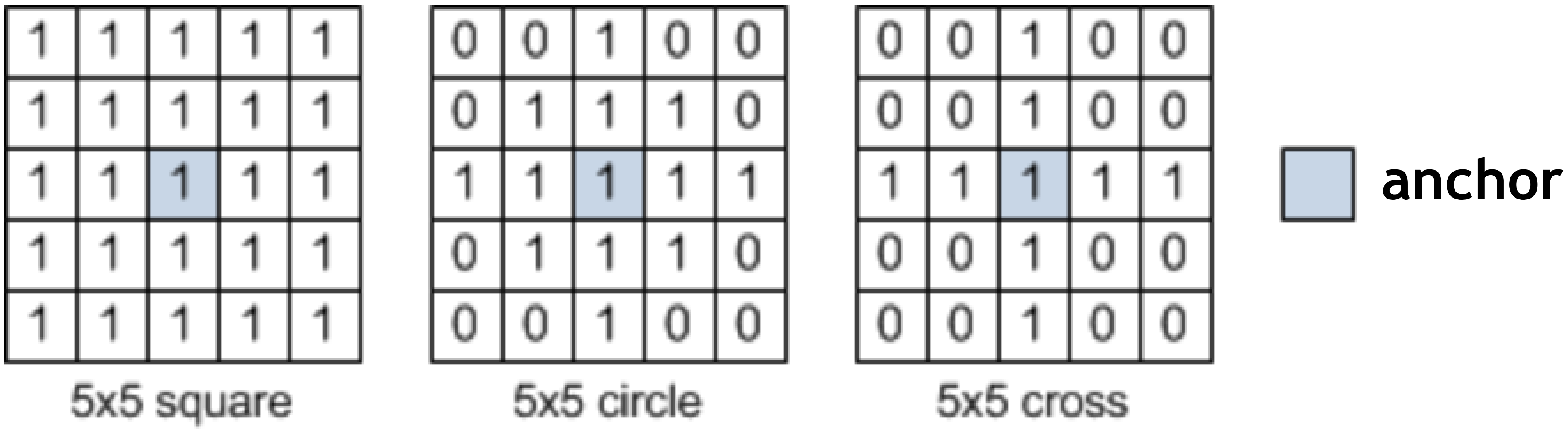


Image kernel

- Visual example: [*https://setosa.io/ev/image-kernels/*](https://setosa.io/ev/image-kernels/)
- **Image kernel** is a small matrix used to apply effects/algorithm.



Kernel multiplication

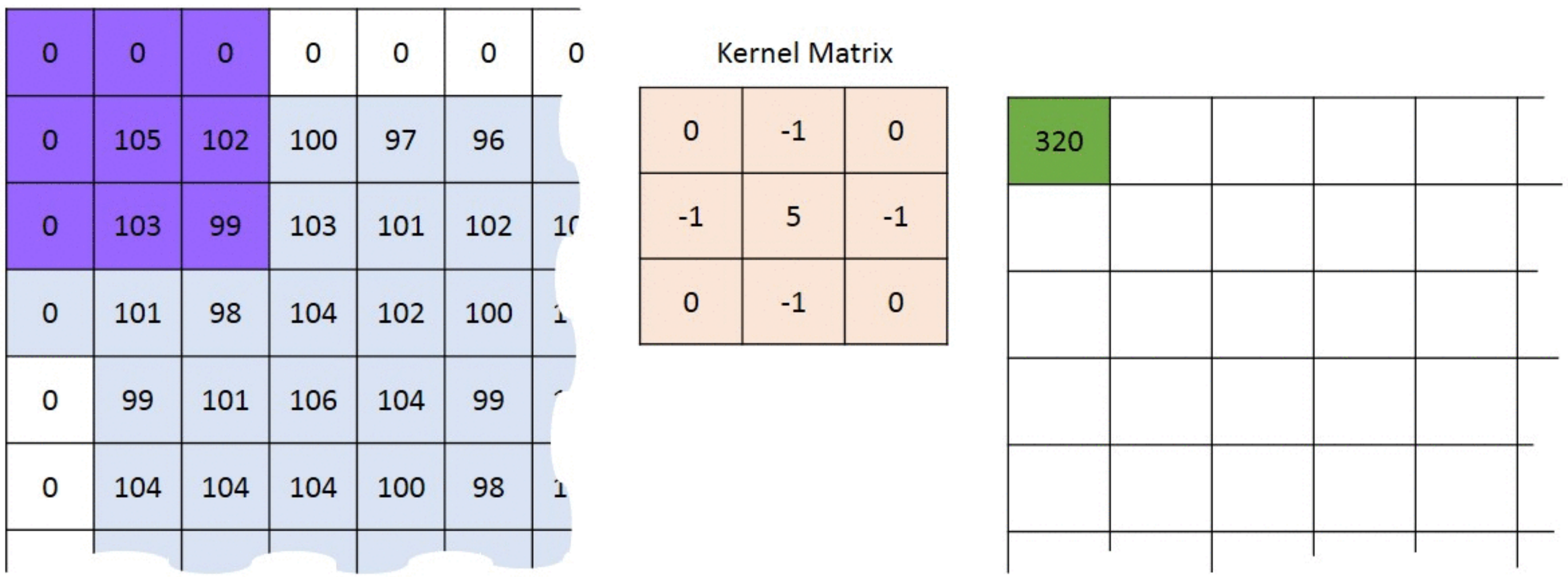


Image Matrix

$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 \quad = 320$$

Output Matrix

Convolution with horizontal and vertical strides = 2

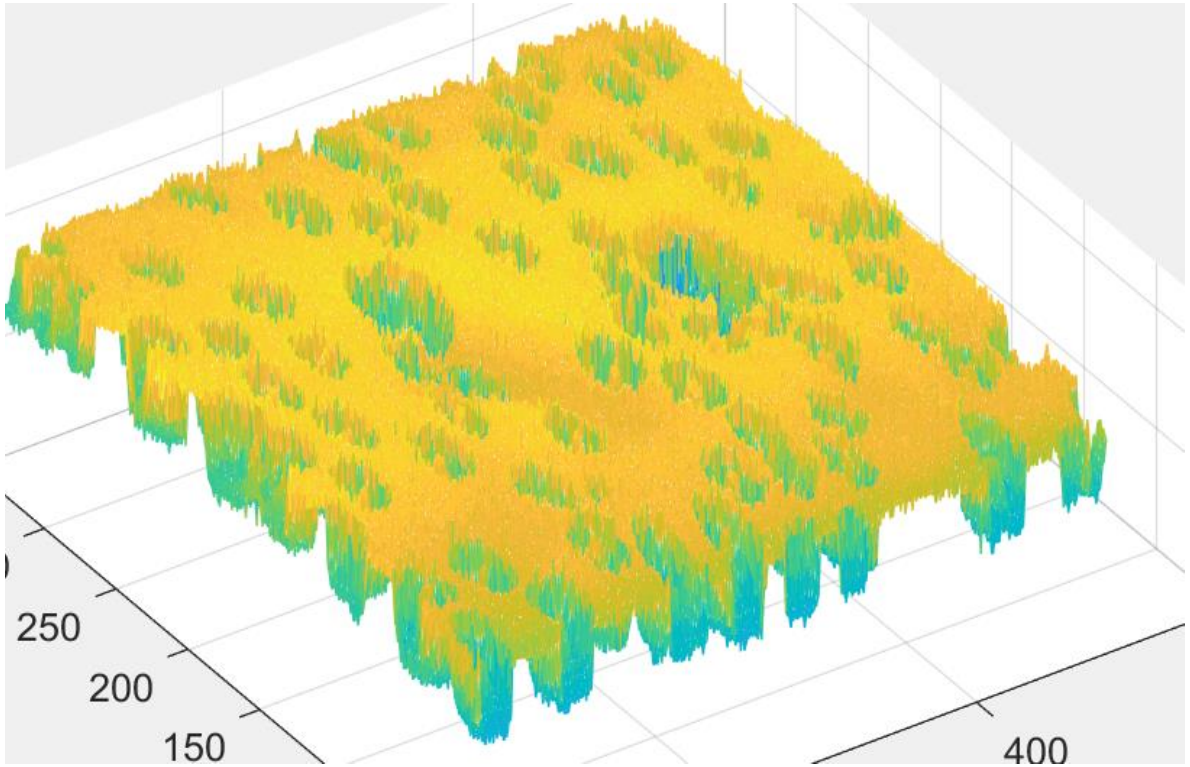
Image smoothing

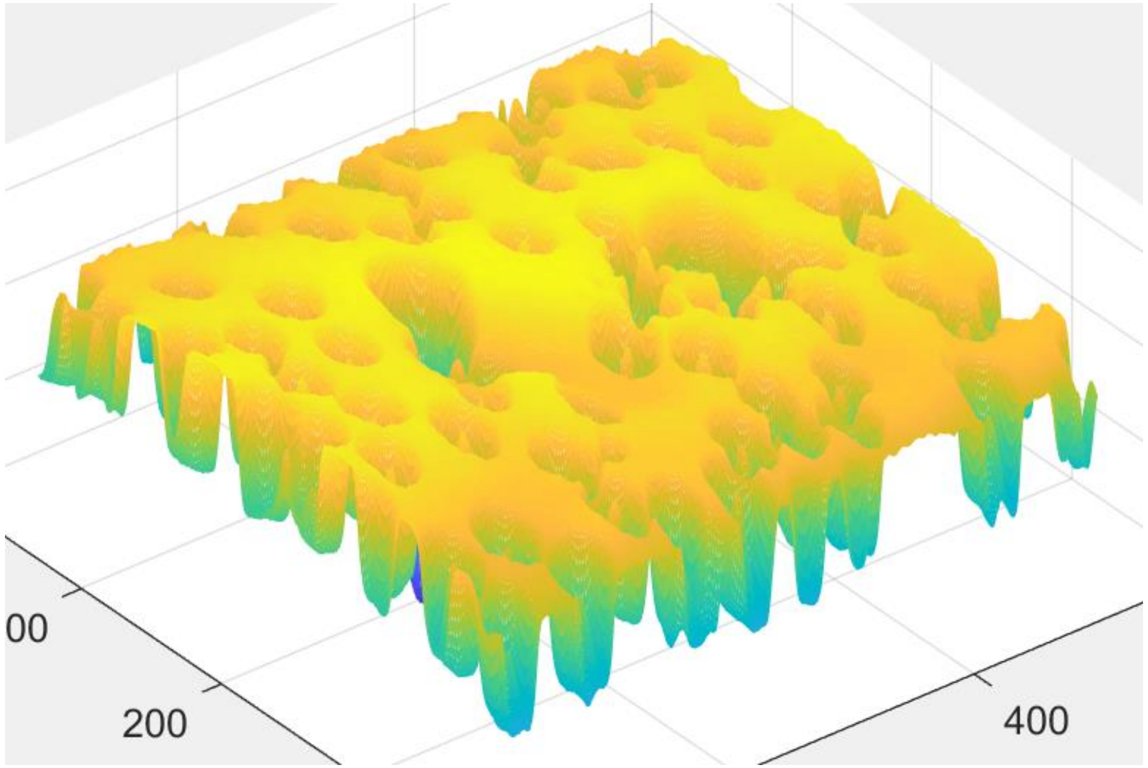
- Opencv functions

```
blur(src,dst,kernel_size,anchor)        → Mean filter, blur edges
medianBlur(src,dst,kernel_size)         → Median Filter, preserve edges
bilateralFilter(src,dst,diameter,sigmaColor,SigmaSpace)
GaussianBlur(src,dst,kernel_size,sigmaX,sigmaY)  → Mean filter with Gaussian kernel
```

```
for(size_t i=1; i<5;i++)
{
    Mat blur_img;
    //medianBlur(gray, blur_img,2*i+1);
    // blur(gray,blur_img,Size(2*i+1,2*i+1), Point(-1,-1));
    GaussianBlur(gray,blur_img,Size(2*i+1,2*i+1),0,0);
    // bilateralFilter(gray,blur_img,i*2,i*2,i*2);
    String window = "kernel = " + to_string(2*i+1) + "x" + to_string(2*i+1);
    imshow( window, blur_img);
}
```

Image smoothing



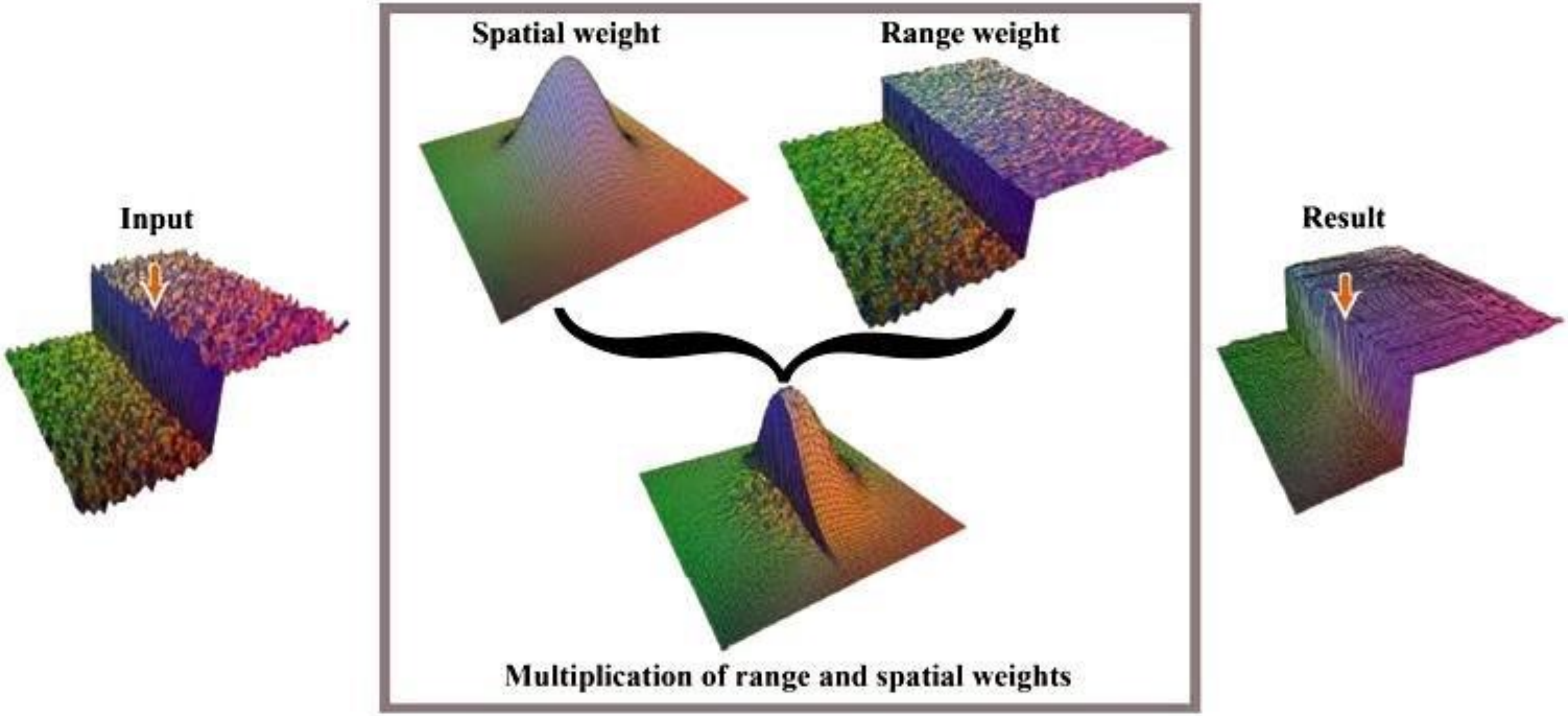


Median filter

1. Search for Neighbourhood values
2. Sort
3. Keep the median

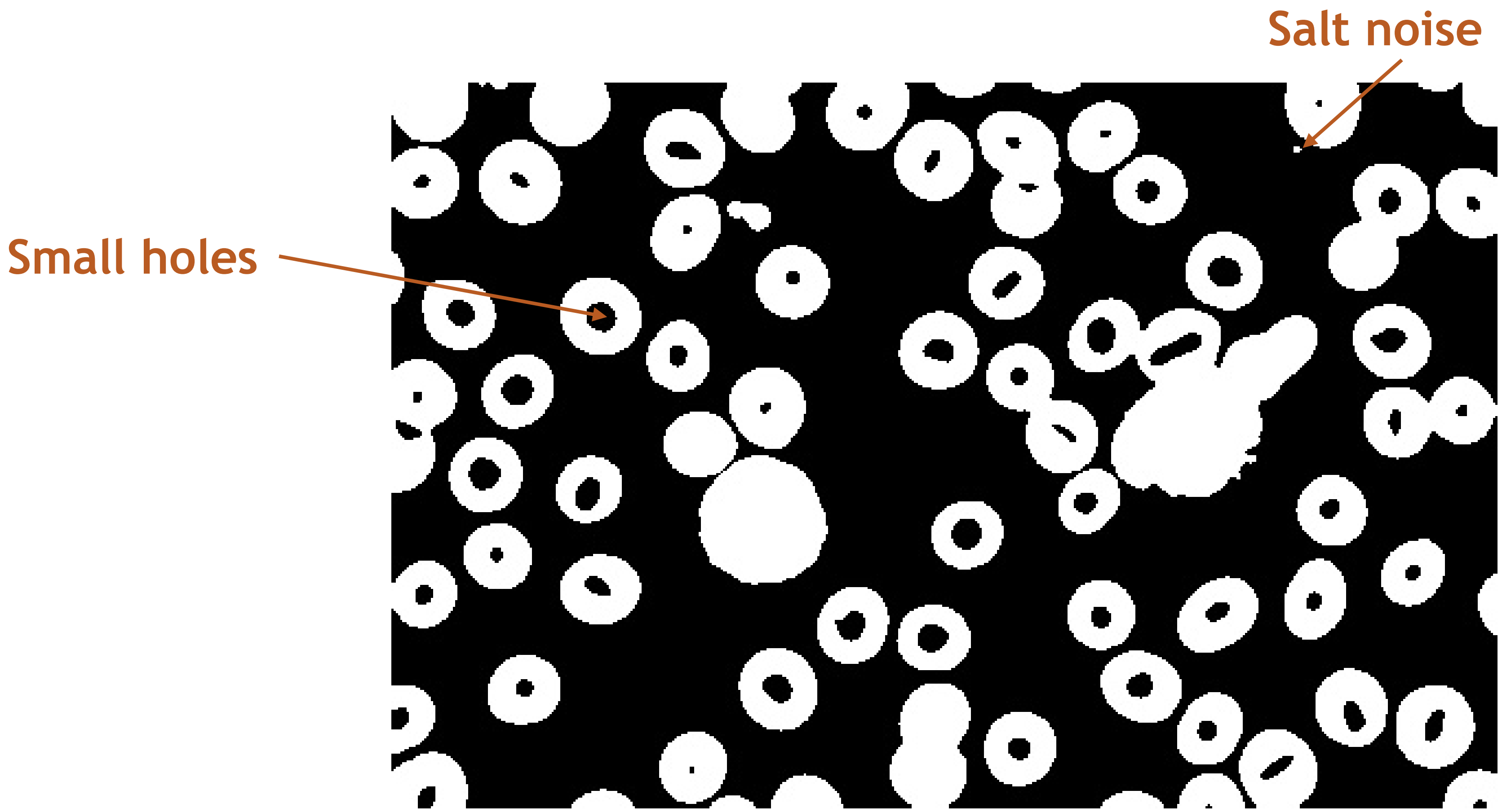
25	151	84	34	62	132
1	224	71	188	178	71
71	153	120	111	238	184
65	61	14	15	26	226
58	144	72	41	94	191
152	66	153	184	18	225

Bilateral filter



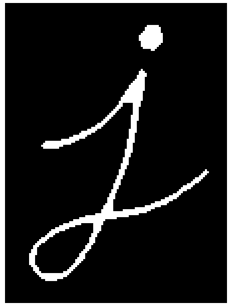
Bilateral filter weights at the central pixel

Problem 2: Small holes / Salt noise

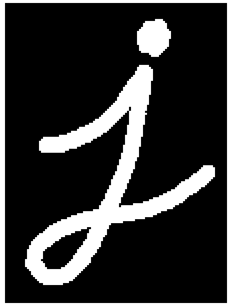


Morphology

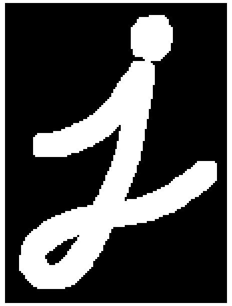
- Morphology: A set of operations that expand or shrink object’s shape
- 2 basic Morphological operations
  - Dilation: Expand the original shapes
  - Erosion: Shrink the original shape



After Erosion



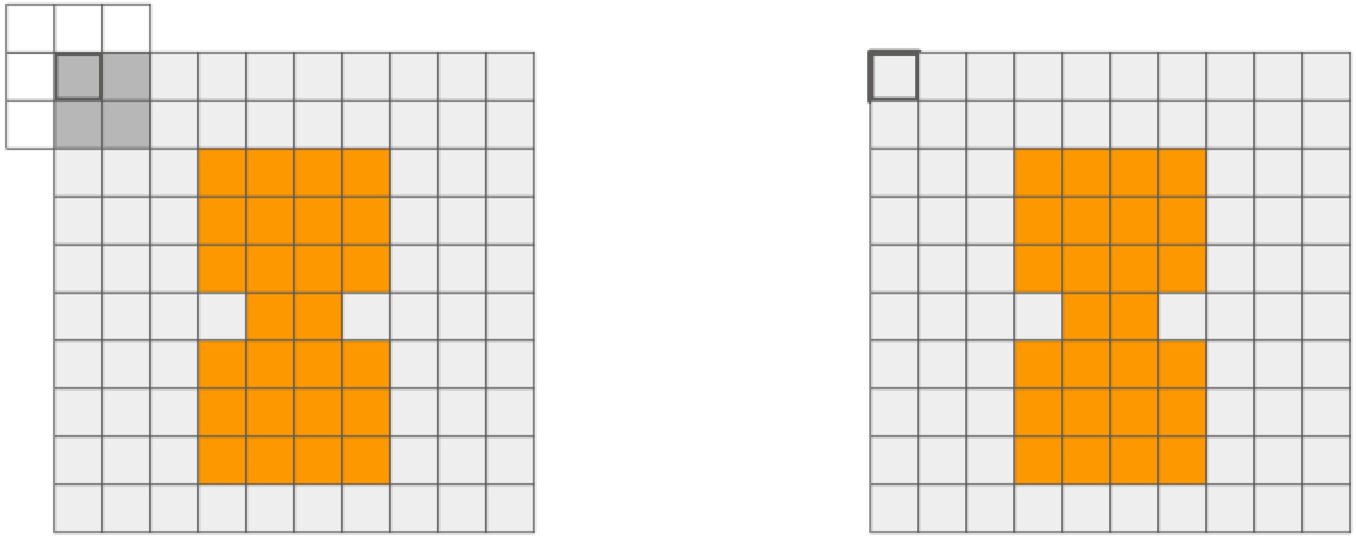
Original object



After Dilation

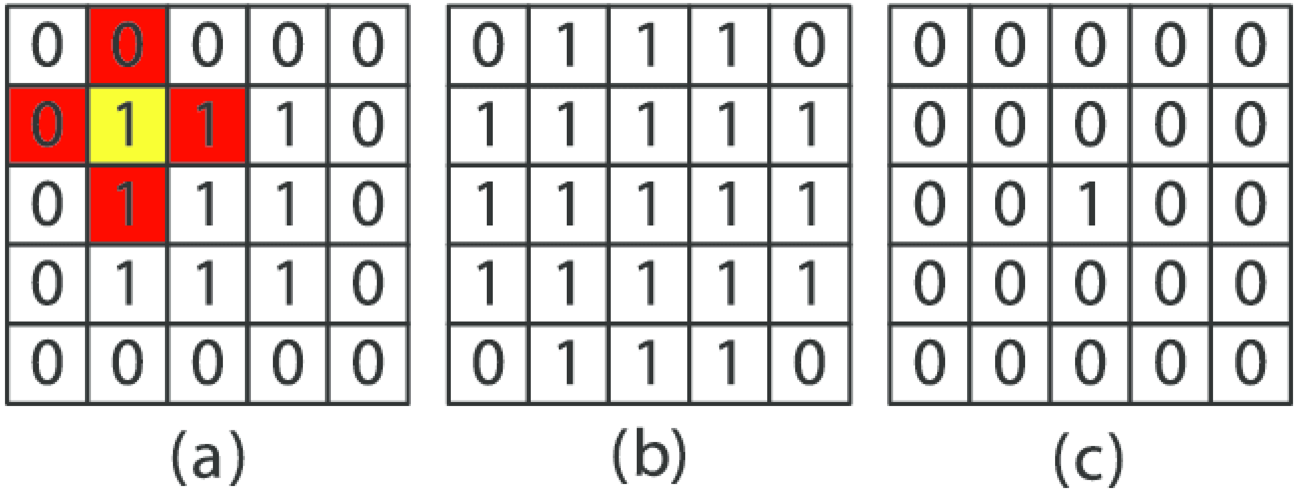
Morphology

- Process of dilation and erosion



Morphology

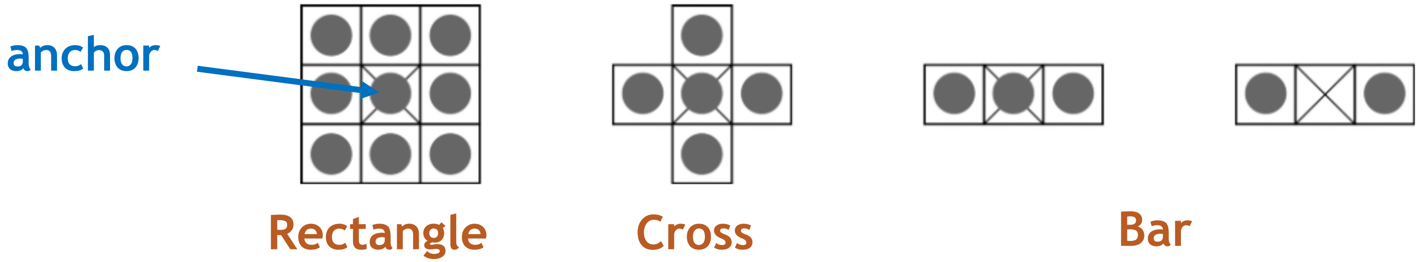
- **Process of dilation and erosion**



Processes of dilation and erosion operations. (a) A 3 × 3 cross shape structural element applied to a 5 × 5 binary image. Each pixel value in the element represented the intensity. (b) Output image obtained from dilation. (c) Output image obtained from erosion.

Morphology

- **Kernel:** matrix/patterns that are used to process images

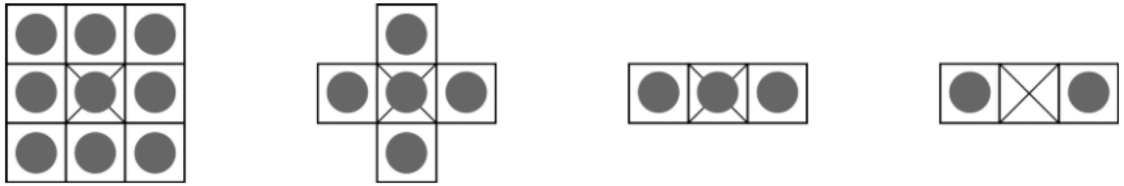


- **OpenCV function:**

```
getStructuringElement(shape, size, anchor)
```

```
// Morphology
Mat element = getStructuringElement( MORPH_RECT, Size(5,5), Point(-1,-1));
```

Morphology



- **Element shape**

Enumerator	
MORPH_RECT Python: cv.MORPH_RECT	a rectangular structuring element: $$E_{ij} = 1$$
MORPH_CROSS Python: cv.MORPH_CROSS	a cross-shaped structuring element: $$E_{ij} = \begin{cases} 1 & \texttt{if } i = \texttt{anchor.y } or j = \texttt{anchor.x} \\ 0 & \texttt{otherwise} \end{cases}$$
MORPH_ELLIPSE Python: cv.MORPH_ELLIPSE	an elliptic structuring element, that is, a filled ellipse inscribed into the rectangle Rect(0, 0, esize.width, 0.esize.height)

Morphology

- Single Operation

```
dilate(src,dst,element,anchor,iteration)
erode(src,dst,element,anchor,iteration)
```

- Advanced Transformation

```
morphologyEx(src,dst,operation,element,anchor,iteration);
```

https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#gaeb1e0c1033e3f6b891a25d0511362aeb

Morphology

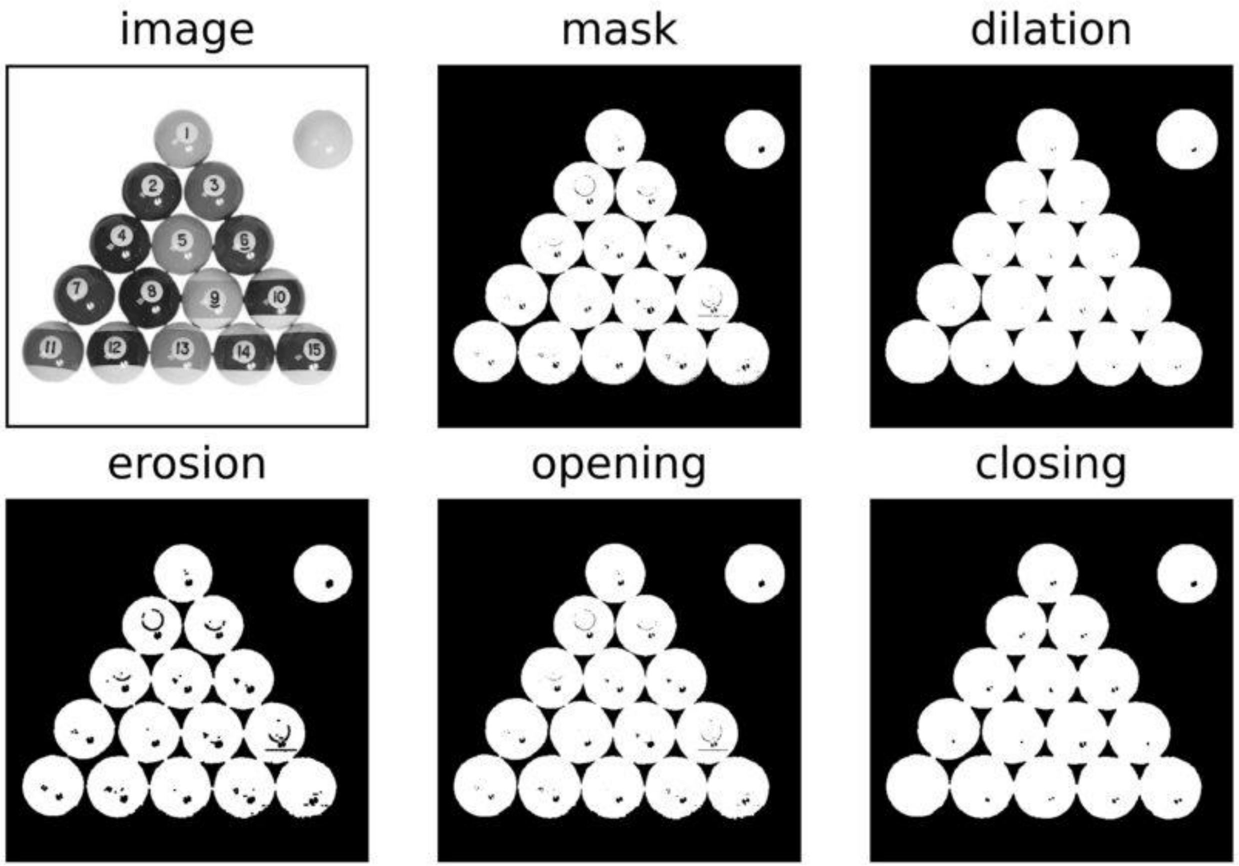
Enumerator	
MORPH_ERODE Python: cv.MORPH_ERODE	see **erode**
MORPH_DILATE Python: cv.MORPH_DILATE	see **dilate**
MORPH_OPEN Python: cv.MORPH_OPEN	an opening operation $\texttt{dst} = \mathrm{open}(\texttt{src}, \texttt{element}) = \mathrm{dilate}(\mathrm{erode}(\texttt{src}, \texttt{element}))$
MORPH_CLOSE Python: cv.MORPH_CLOSE	a closing operation $\texttt{dst} = \mathrm{close}(\texttt{src}, \texttt{element}) = \mathrm{erode}(\mathrm{dilate}(\texttt{src}, \texttt{element}))$
MORPH_GRADIENT Python: cv.MORPH_GRADIENT	a morphological gradient $\texttt{dst} = \mathrm{morph\_grad}(\texttt{src}, \texttt{element}) = \mathrm{dilate}(\texttt{src}, \texttt{element}) - \mathrm{erode}(\texttt{src}, \texttt{element})$
MORPH_TOPHAT Python: cv.MORPH_TOPHAT	"top hat" $\texttt{dst} = \mathrm{tophat}(\texttt{src}, \texttt{element}) = \texttt{src} - \mathrm{open}(\texttt{src}, \texttt{element})$
MORPH_BLACKHAT Python: cv.MORPH_BLACKHAT	"black hat" $\texttt{dst} = \mathrm{blackhat}(\texttt{src}, \texttt{element}) = \mathrm{close}(\texttt{src}, \texttt{element}) - \texttt{src}$
MORPH_HITMISS Python: cv.MORPH_HITMISS	"hit or miss" .- Only supported for CV_8UC1 binary images. A tutorial can be found in the documentation

https://docs.opencv.org/master/d4/d86/group__imgproc__filter.html#ga7be549266bad7b2e6a04db49827f9f32

Morphology

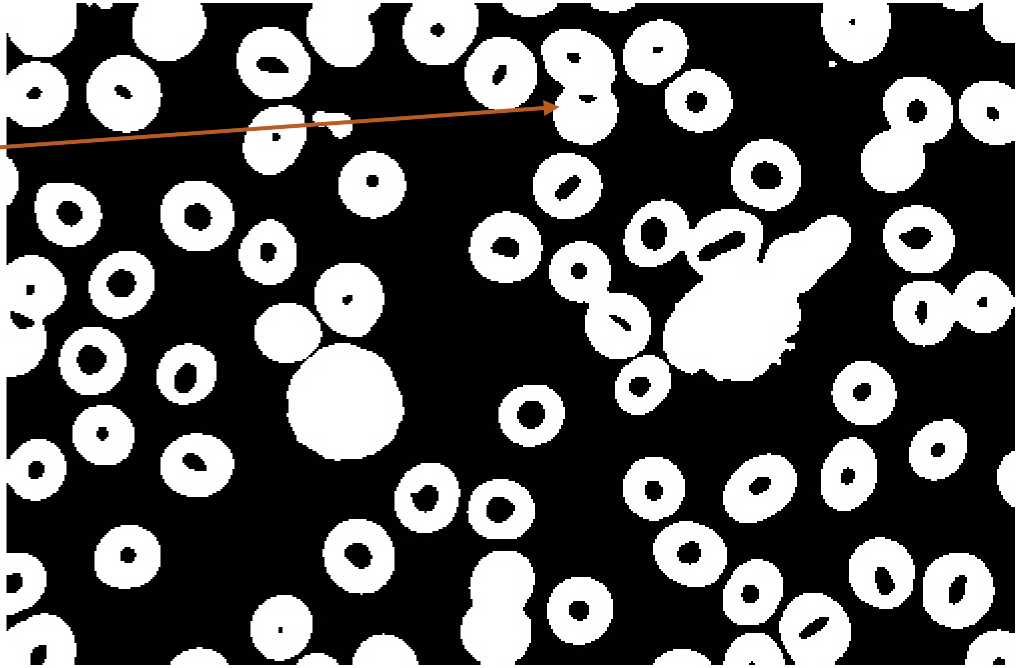
- Opening is defined as an **erosion followed by a dilation** *using the* ***same structuring element*** *for both operations*

- Closing is defined as a **dilation followed by an erosion** *using the* ***same structuring element*** *for both operations.*



Problem 3: Object cluster/occlusion

Cell stick to each other



Distance transform

- Distance transform convert matrix of pixel intensity values into distance map
  - Each pixel now show the shortest distance from itself to the nearest black area
  - Only apply for binary image

http://www.cmm.mines-paristech.fr/~marcoteg/cv/publi_pdf/MM_refs/1986_Borgefors_distance.pdf

Distance transform

- Opencv Function:

```
distanceTranform(src,dst,distanceType,markSize)
```

Distance Type
type of the distance transformation to be applied

Mark Size
Size of the distance transform mask/kernel

```
// distance transform
Mat dist;
distanceTransform(gray,dist,DIST_L2,3);
// normalize to rang of [0 1]
normalize(dist, dist, 0, 1.0, NORM_MINMAX);
imshow("Distance Transform Image", dist);
```

Distance transform

- Distance type

<table>
<tr><th colspan="2">Enumerator</th></tr>
<tr><td>DIST_USER
Python: cv.DIST_USER</td><td>User defined distance.</td></tr>
<tr><td>DIST_L1
Python: cv.DIST_L1</td><td>distance = |x1-x2| + |y1-y2|</td></tr>
<tr><td>DIST_L2
Python: cv.DIST_L2</td><td>the simple euclidean distance</td></tr>
<tr><td>DIST_C
Python: cv.DIST_C</td><td>distance = max(|x1-x2|,|y1-y2|)</td></tr>
<tr><td>DIST_L12
Python: cv.DIST_L12</td><td>L1-L2 metric: distance = 2(sqrt(1+x*x/2) - 1))</td></tr>
<tr><td>DIST_FAIR
Python: cv.DIST_FAIR</td><td>distance = c^2(|x|/c-log(1+|x|/c)), c = 1.3998</td></tr>
<tr><td>DIST_WELSCH
Python: cv.DIST_WELSCH</td><td>distance = c^2/2(1-exp(-(x/c)^2)), c = 2.9846</td></tr>
<tr><td>DIST_HUBER
Python: cv.DIST_HUBER</td><td>distance = |x|<c ? x^2/2 : c(|x|-c/2), c=1.345</td></tr>
</table>

Distance transform

- Mask size

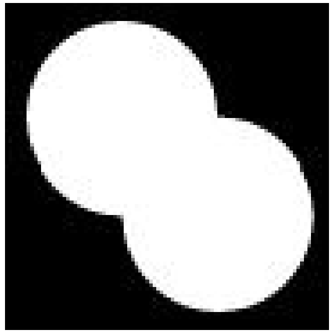
+b	+a	+b
+a	0	+a
+b	+a	+b

3 x 3 mask

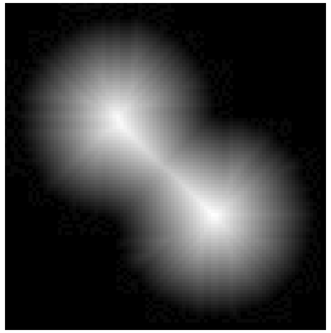
-	+c	-	+c	-
+c	+b	+a	+b	+c
-	+a	0	+a	-
+c	+b	+a	+b	+c
-	+c	-	+c	-

5 x 5 mask

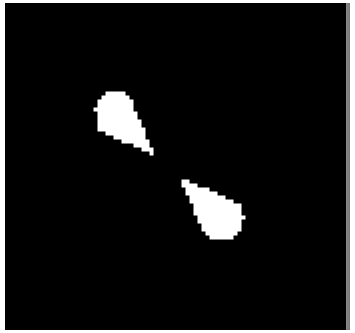
Distance transform

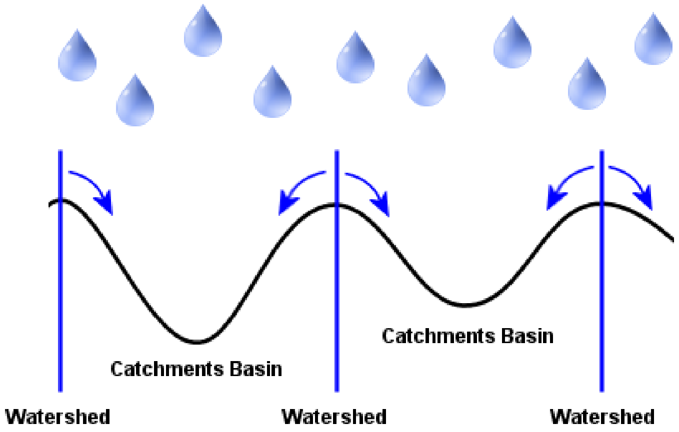


Distance Transform

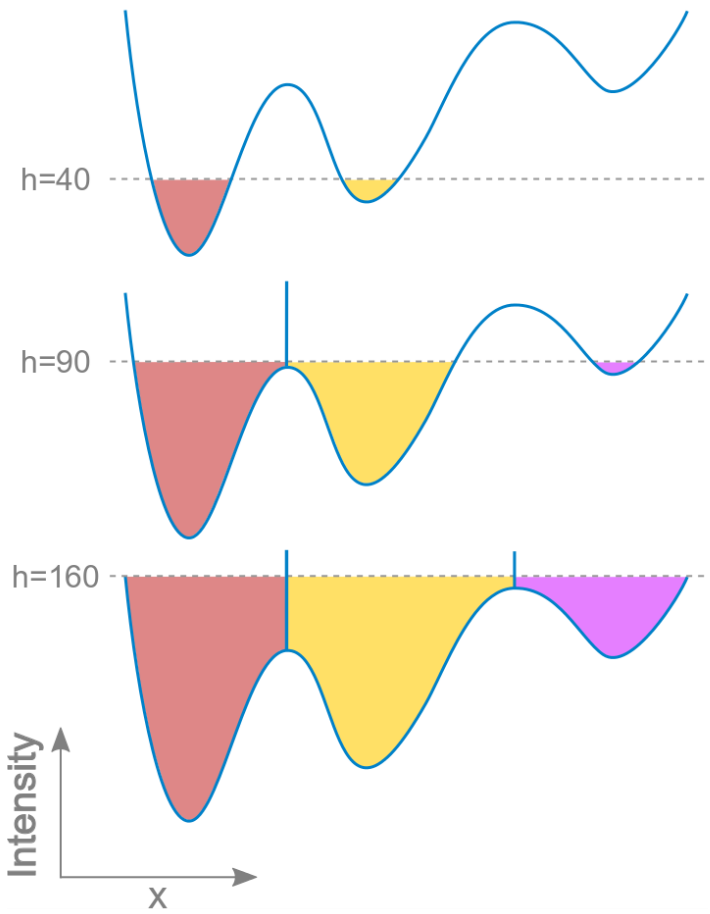


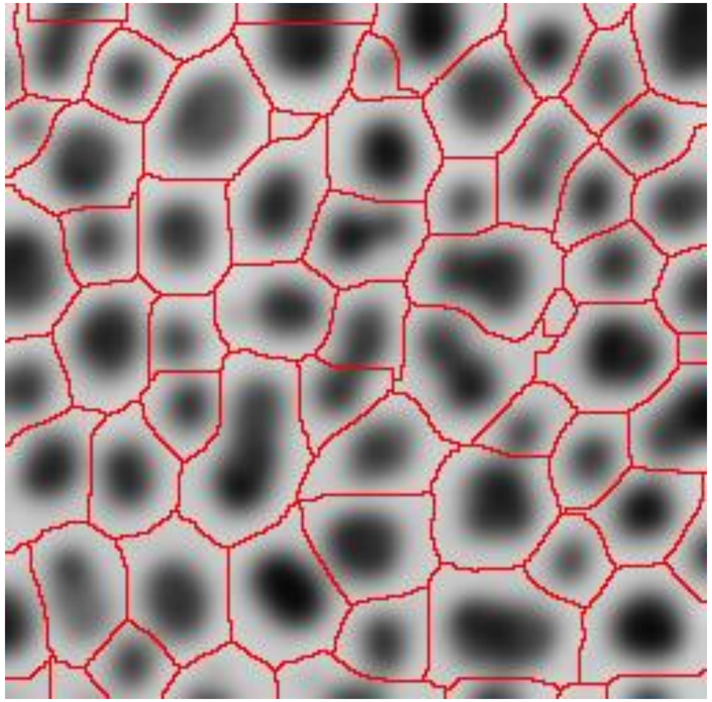
- Normalized to [0 1]
- Threshold





Watershed





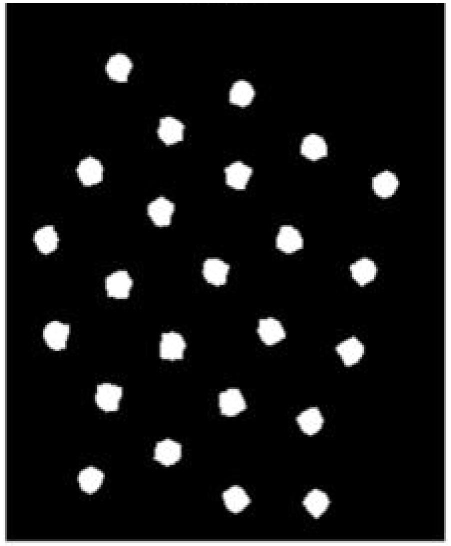
https://docs.opencv.org/3.4/d2/dbd/tutorial_distance_transform.html

Watershed

- Opencv Function:

```
watershed(image, markers)
```

image



Markers
(before)

Result