40AR015 – Exercises

Author: Filip Kadlec

E1-1 Why the result of applying the filter LoG and after the zero crossing, always returns closed contours?

Zero crossing always lies on a closed contour. This is sometimes being compared to contour lines on an elevation map, which are also always closed: zero-crossing are contours with "elevation" equal to zero, that means the value of the LoG picture does not change on one contour. When we compare zero-crossing problem with the topographic map: LoG filtered image represents the "heights" with positive and negative "elevations" on both sides of the zero-crossing contours.

There might be an exception – curve goes of the image. Output of the zero crossing is usually a binary image with a one-pixel thickness lines, which represent zero crossing points.

E1-2 Why the Sobel filter obtains wider contours than the zero-crossing filter

Zero crossing returns contours with only one-pixel thickness showing the position of the zero crossing points as stated in the previous exercise. Sobel on the other hand has two-pixel border, which is given by the structure of the filter. Size of the matrix of Sobel filter is 3x3 and if there is an edge (vertical, horizontal), that is represented by two columns (rows) with big difference of values. Pixels of these two columns (rows) both return high (low) values when filtered by Sobel filter, since if one of these columns is being filtered, the other one is always present in the matrix multiplication.

It is much easier to explain the principle of Sobel filter and why does it return two-pixel edge:

100	100	100	20	20	20
100	100	100	20	20	20
100	100	100	20	20	20
100	100	100	20	20	20
100	100	100	20	20	20
100	100	100	20	20	20

A)

−1	0	1
−2	0	2
−1	0	1

B)

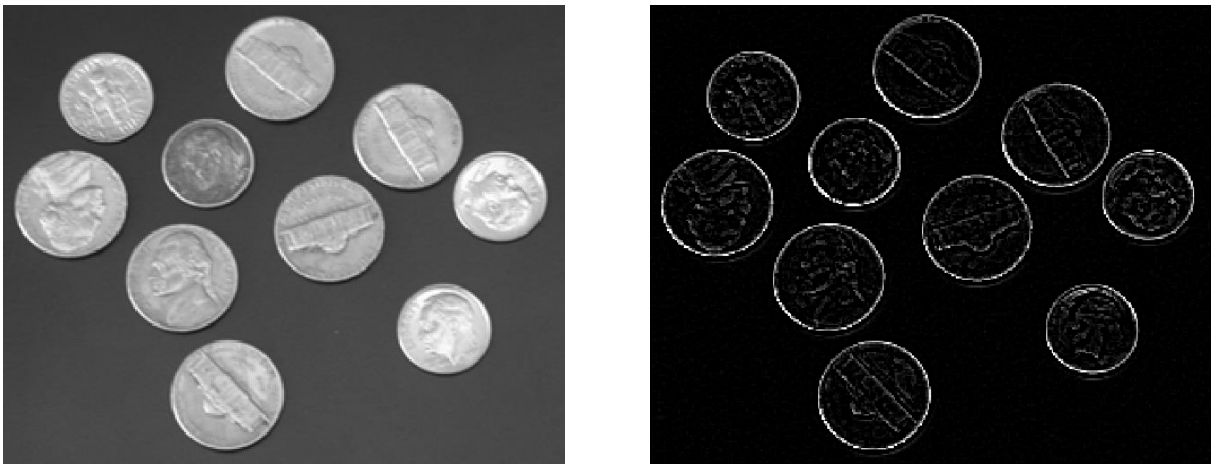
Figure 1: A) Matrix containing vertical edge. B) Sobel vertical filter

300	0	−240	−240	0	−60
400	0	−320	−320	0	−80
400	0	−320	−320	0	−80
400	0	−320	−320	0	−80
400	0	−320	−320	0	−80
300	0	−240	−240	0	−60

Figure 2: Result of Sobel filtering of contents of Figure 1

E1-3 Compare LoG & Laplacian after median filtering

These two methods are not returning the same result. Applying Laplacian after median has the main advantage of the median filter reducing noise in the picture first. On the other hand, this method is not capable of a quality edge detection in complex images (if edges are too complex or too close to each other, median blurres the area), but LoG is. One more disadvantage is, that median is using non-linear operations, which means that he is slower in larger images.
LoG is not dealing with noise, so it detects a lot of undesirable edges. On the other hand, **if the sigma is set correctly**, the results might overcome the "Laplacian after median" filtering:



A) *B)*

Figure 3: A) Original image. B) Result after applying Laplacian after median filter

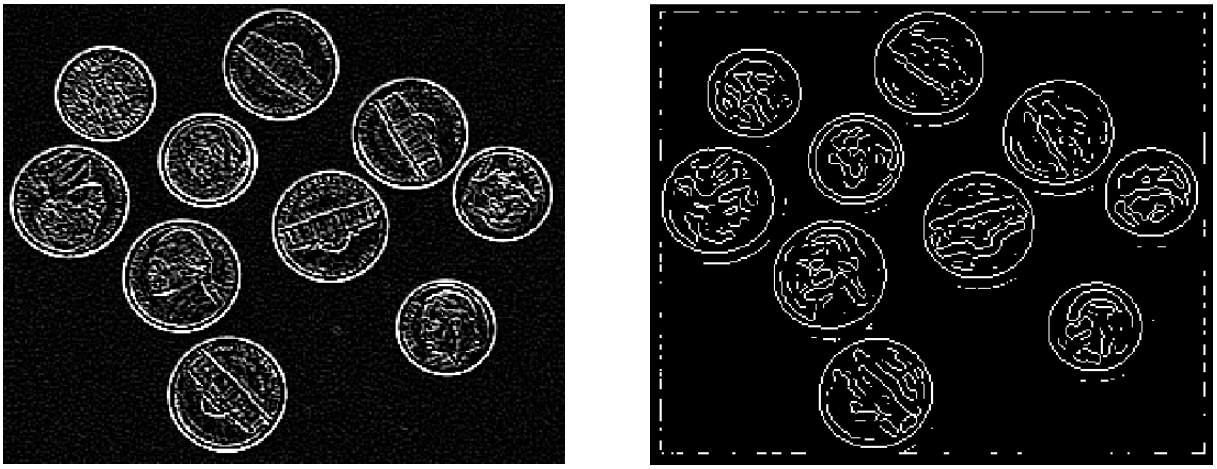
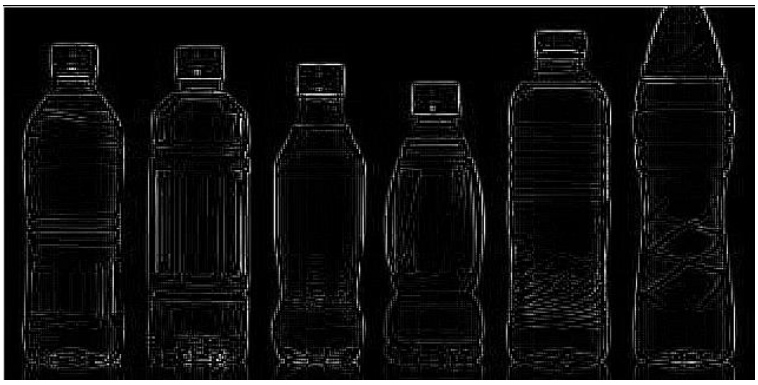


Figure 4: Results applying LoG on original image using defferent sigmas

E2-1 Given figure of bottles, extract the contour using the following functions: Laplacian, Roberts and Sobel detectors. Extract also the contours using a morphological operator. ¿Which of them obtain the contours with only a pixel and eliminating the noisy points?

At first it is needed to extract contours using mentioned functions. That is done by defining specific filters that represent mentioned functions and will work as contour detectors. For this task the function *imfilter()* Matlab provides is a good fit apart from the last method for contour detection- use of morphological operations. There the functions *imdilate()* and *imerode()* will do the work.



A) Laplacian

B) Roberts

B) Sobel

D) Morphological operations

Figure 5: Extracted contours using different detectors

The answer to the question "Which of them obtain the contours with only a pixel and eliminating the noisy points" is: **D) Morphological operations**. Only the morphological operations ensure that the noisy points are being eliminated and the contours have one-pixel width by using:

$$\beta(A) = A - (A \otimes B)$$

E2-2 For each one of the bottle contours, obtain the following features: perimeter, area, compacity, position of the bottles and orientation of the bottles.

Before obtaining given features, it is needed to segment each bottle in the image. That is done by choosing the best result from the previous exercise- extracted contours by morphological operations. Then transferring this image to binary form with optimal threshold.

Figure 6: Binarized image of obtained contours by morph. Operations

After binarizing the image, the hole filling takes place. After that, any small blobs that were not thresholded before are removed.

Figure 7: Hole filling and small blobs removing

It is noticeable that image has a lot of imperfections at the moment, so additional operations (erosion, dilation) are needed. After that, the image is suitable for objects labelling and perimeter extraction.

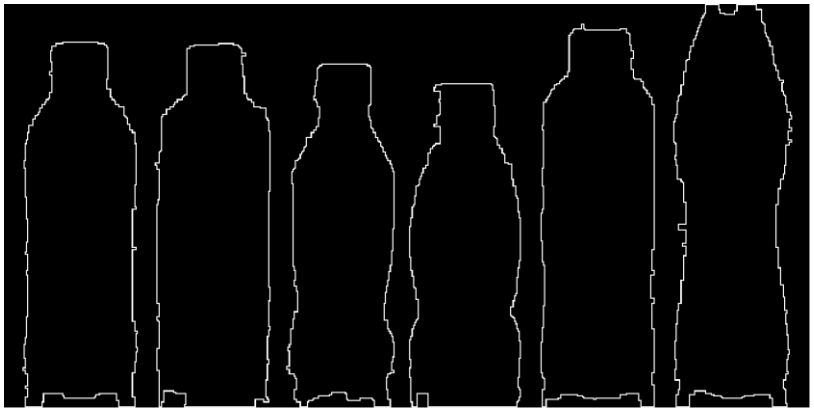


Figure 8: Perimeter of bottles in the image

The wanted features, such as perimeter, area, compacity, position of the bottles and orientation of the bottles was at first computed manually, then using *regionprops()* function. The results were close to each other, but the built-in matlab function optimizes the computation and ignores some imperfections in the image, so the final result is presented using this function.

Figure 9: Labeled objects in image

Labels go as follows: 1, 2, 3, 4, 5, 6. This information is needed for the following table of results.

	1	2	3	4	5	6
Perimeter	789	780	732	715	810	886
Area	23893	24858	20028	20624	26065	26524
Compacity	26.0243	24.4884	26.7798	24.7883	25.1719	29.6497
Position	[63 ; 181.5]	[172 ; 182]	[277 ; 190.5]	[379.5 ; 198.5]	[488.5 ; 174]	[599.5 ; 166]
Orientation	89.93	89.71	89.43	89.83	89.74	-89.99

A note regarding position of objects in the table above- first coordinate is horizontal from the left to right, the second coordinate is vertical from top to bottom.

E2-3 Using the features of point 2, can you identify univocally each one of the bottles?

That depends on the specification of the problem. The most useful feature extracted in the previous exercise is **Compacity**. Meaning, bottles can be anywhere in the image and the orientation can by arbitrary (the position and orientation may vary). Also bottles might be captured further or closer to the camera, so perimeter and area would also vary. But if the images of bottles were taken in the proper environment (it would be possible to segment the bottles), it should be possible to identify the bottles based on the compacity, since every bottle has its own, there are not two same compacities for two different bottles and also the compacity stays the same no matter what the other parameters are.
It is also noteworthy, that if the company wanted to identify the bottles more efficiently and reliably, the images of bottles should be taken in a more suitable environment- environment where bottles

does not have almost the same colour as their background and most importantly, where there will **not be a reflection** of the bottles. That is the source of the most inaccuracies in the computation of the features.

E3 Which are the computer vision operations required to pass from the image of a) to b)

The goal of exercise number three is to pass from image a) to image b) in figure 10 by using computer vision operations.

a)												b)											
0	0	0	70	70	70	0	0	70	70	0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	70	70	70	0	0	70	70	0	0	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	70	70	70	0	0	70	70	0	0	0	1	1	0	0	1	0	0	0	0	0	0
0	0	50	20	20	20	50	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
0	50	20	20	20	40	20	50	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
0	50	20	20	40	20	20	50	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
0	50	20	40	20	20	20	50	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
0	50	40	20	20	20	20	50	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
0	0	50	20	20	20	50	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
0	50	20	20	20	20	20	50	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0
0	50	50	50	50	50	50	50	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0

Figure 10: Input and output image of exercise number three

Computer vision operations used in this exercise were strictly chosen to obtain the final image. There is no practical use of these operations in this particular order. Final image obtained by this sequence of operations is not exactly the same as the one required in assignment, although it is the closest one obtained. Each operation and its structuring elements are shown in the following figure.

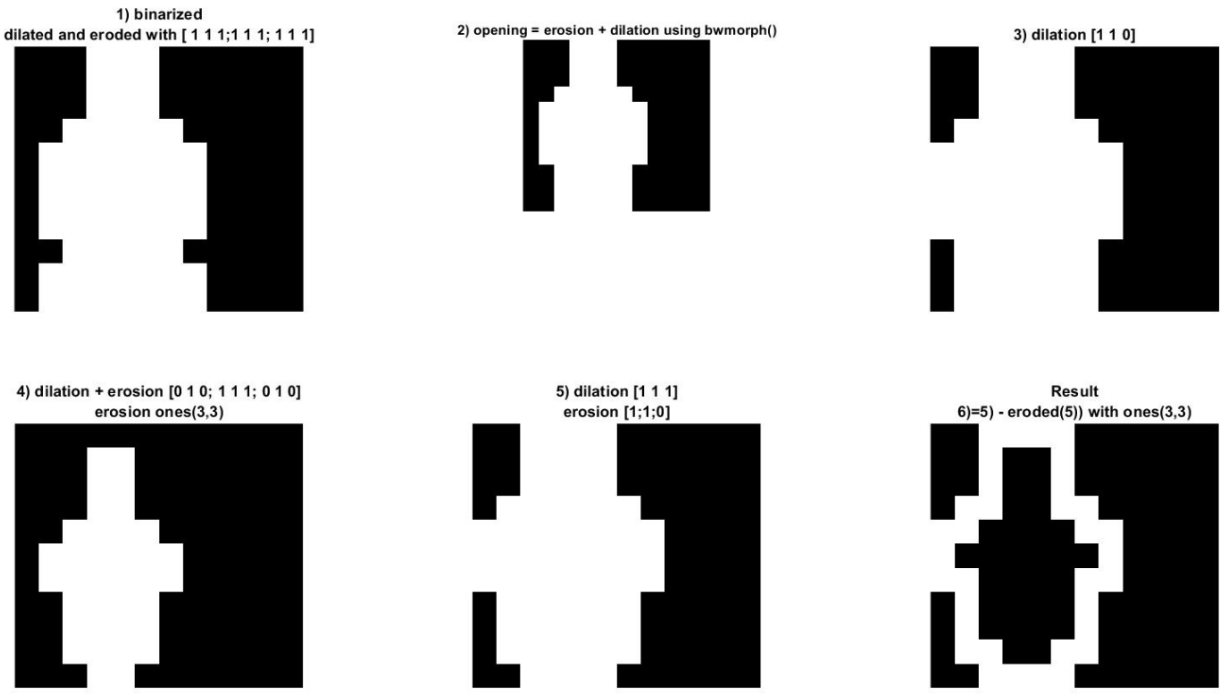


Figure 11: Sequence of operations used in exercise 3