TDT4195: Visual Computing Fundamentals

Digital Image Processing - Assignment 3

November 10, 2017

Aleksander Rognhaugen
Department of Computer Science
Norwegian University of Science and Technology (NTNU)

- **Delivery deadline: November 24, 2017** by 23:00.
- **This assignment counts towards 6 % of your final grade.**
- You can work on your own or in groups of two people.
- Deliver your solution on *Blackboard* before the deadline.
- Please upload your report as a PDF file, and package your code into a single ZIP archive. Not following this format may result in a score deduction.
- The programming tasks may be completed in the programming language of your choice, however, it might be a good idea to select one that supports matrix and image processing operations, e.g. MATLAB or Python with NumPy. *The lab computers at ITS-015 support Python and MATLAB.*
- The delivered code is taken into account with the evaluation. Ensure your code is documented and as readable as possible.
- For each programming task you need to give a brief explanation of what you did, answer any questions in the task text, and show any results, e.g. images, in the report.
- In this assignment, you can use existing implementations of morphological operations, such as `scipy.ndimage.binary_closing()` from SciPy[1] (Python) and `imclose()` from MATLAB[2].

Objective: Gain experience with (a) how an image can be segmented into a set of disjoint regions, (b) how mathematical morphology can be used to manipulate the contents of images, and (c) how basic shape recognition can be accomplished.

[1] SciPy: `https://docs.scipy.org/doc/scipy-1.0.0/reference/ndimage.html#morphology`
[2] MATLAB: `https://se.mathworks.com/help/images/morphological-filtering.html`

Task 1: Theory Questions [1 point]

a) **[0.2 points]** What are the main aims of segmentation in computer vision? Give at least three reasons why segmentation is difficult.

b) **[0.4 points]** The Hough transform is a technique for identifying probable instances of a particular shape within an image.

Consider the 14 images in Figure 2. The seven images in the left-hand column are binary images, while the seven images in the right-hand column are images of the Hough parameter space for *straight lines*.

For each of the seven binary images in the left-hand column (**A**-**G**), indicate which image in the right-hand column (**1**-**7**) is the result of applying the Hough transform for straight lines. Explain your reasoning.

Failure to provide an explanation will result in a severe score deduction.

c) **[0.2 points]** Are morphological operations linear? Explain your reasoning.

d) **[0.2 points]** Determine the *erosion* $\boldsymbol{A} \ominus \boldsymbol{B}$ of the 6×5 binary image in Figure 1 (a). Use the 2×2 structuring element next to the image. The reference pixel is indicated by a circle.

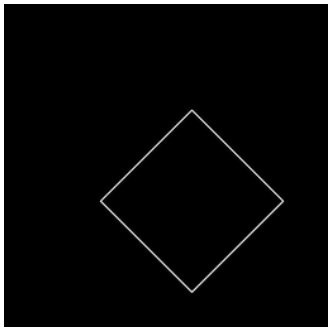
1	0	1	1	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1
1	0	1	1	1
0	1	0	1	1

(a)

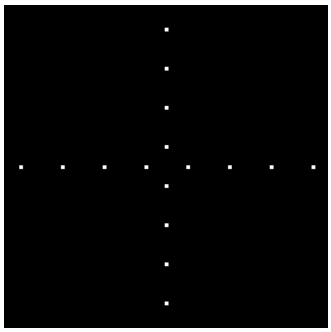
●	1
1	1

(b)

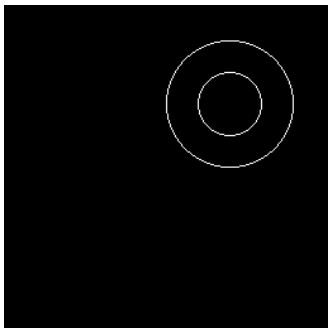
Figure 1: A binary image and a structuring element. The foreground is coloured white and given the symbol 1. The background has the symbol 0 and is coloured black. The reference pixel in the structuring element is indicated by a black circle.



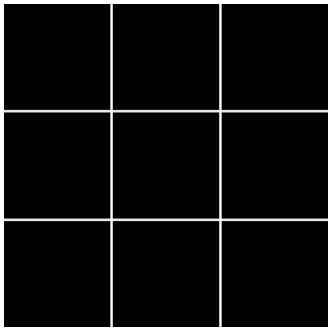
A



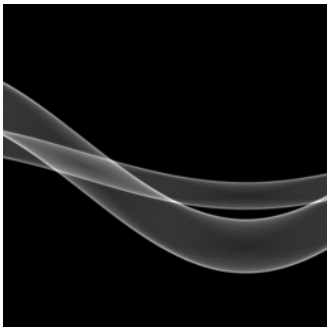
B



C

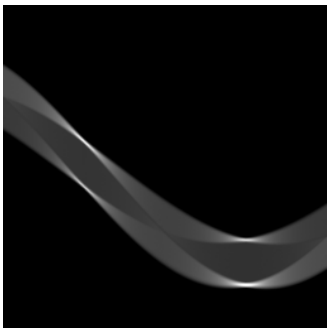


D



1

2



3

4

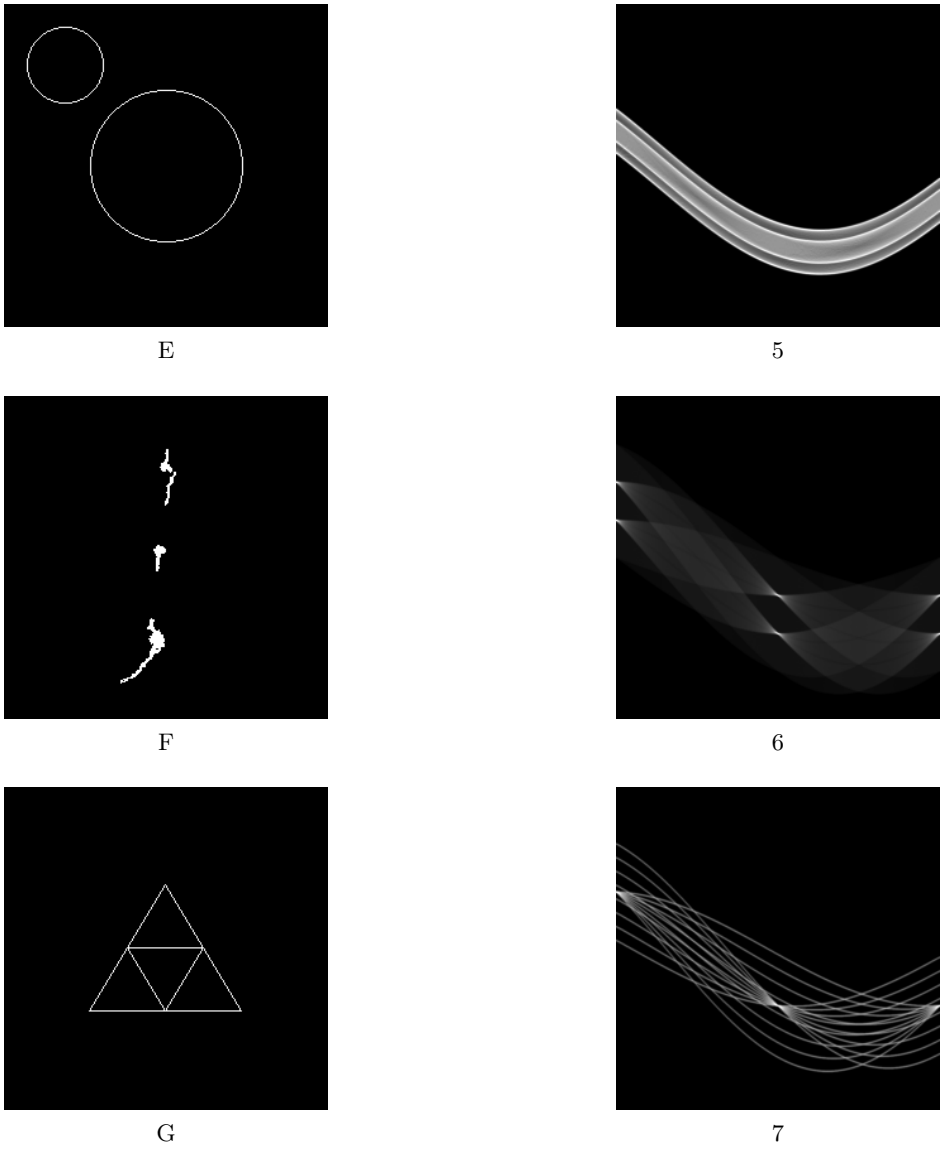


Figure 2: The left-hand column contains binary images, while the right-hand column contains the parameter space of images created using the Hough transform for straight lines. The contrast has been slightly adjusted for some of the parameter space images to improve visibility. The ordering has been randomised. For the images in the right-hand column, the vertical axis indicates the distance from the origin, while the horizontal axis represents the angle from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$, in radians.

Task 2: Region Growing [1 point]

Region growing is a region-based segmentation algorithm that uses a set of seed points and a homogeneity criteria $H(R_k)$ to perform segmentation. For each seed point $s^{(k)}$, a region is grown by inspecting neighbouring pixels and evaluating whether or not to include them in region R_i using the homogeneity criteria $H(R_k)$. The neighbouring pixels that are currently being evaluated are typically called *candidate* pixels. The growing of a region stops when there are no more candidate pixels to inspect.

One simple homogeneity criteria is a *threshold*, where the threshold defines the maximum difference in intensity between the seed point and the current candidate pixel. This can be expressed mathematically as: $|\boldsymbol{I}_{i,j} - s^{(k)}| < \mathbf{T}$ for a threshold $\mathbf{T}$, a seed point intensity value $s^{(k)}$, and a candidate pixel intensity value $\boldsymbol{I}_{i,j}$ in image $\boldsymbol{I}$. That is, the pixel at location (i, j) is accepted into region R_k associated with $s^{(k)}$ if the aforementioned statement is true.

Segmenting the image in Figure 3 (a) using region growing with four manually selected seed points can be seen in Figure 3 (b).

(a) (b)

Figure 3: (a) is an X-ray image of a defective weld. In (b) the image has been segmented using *region growing* with four manually selected seed points.

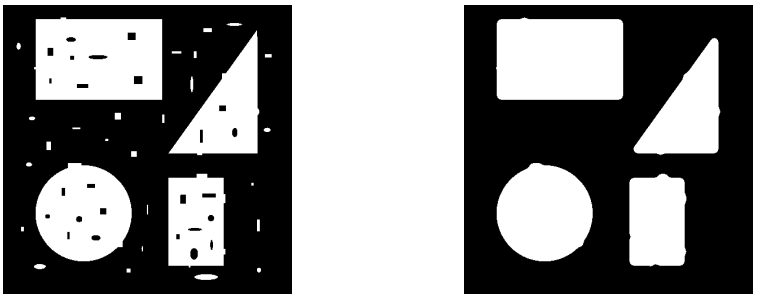
a) **[1 point]** Implement a function that segments a greyscale image using an arbitrary number of seed point coordinates. You *must* implement the region growing approach outlined above from scratch.

When growing a region around a particular seed point, you may expand your set of candidate pixels using either a von Neumann neighbourhood (4-connectedness) or a Moore neighbourhood (8-connectedness).

i) Use your region growing implementation to segment one or more appropriate images. Show the before and after image in your report. Additionally, please write down which seed point coordinates you selected.

Hint: Seed point coordinates can be picked manually by, for example, picking locations of pixels in the original image which are clearly within the region(s) you want to segment.

Task 3: Noise Removal in Binary Images [0.5 points]



(a) (b)

Figure 4: (a) illustrates a noisy binary image. This could, for instance, be the result of having applied thresholding on a noisy greyscale image. An almost noise free version can be seen in (b).

a) **[0.5 points]** Use what you know about *erosion*, *dilation*, *opening*, and *closing* to remove the noisy elements from the image in Figure 4 (a). Your result should look something like the one in Figure 4 (b).

   i) In your own words, explain your noise removal process.

   ii) Display the result in your report.

   Hint: Ensure that the image is a binary image before applying applying morphological operations. This is easily done by thresholding.

Task 4: Boundary Extraction [0.5 points]

There are many useful applications of mathematical morphology, and in this task we will look at one that can be built using only erosion.

One way to extract edge information from binary images can be seen in Equation 1.

$$\boldsymbol{A}_{\text{boundary}} = \boldsymbol{A} - (\boldsymbol{A} \ominus \boldsymbol{B}) \tag{1}$$

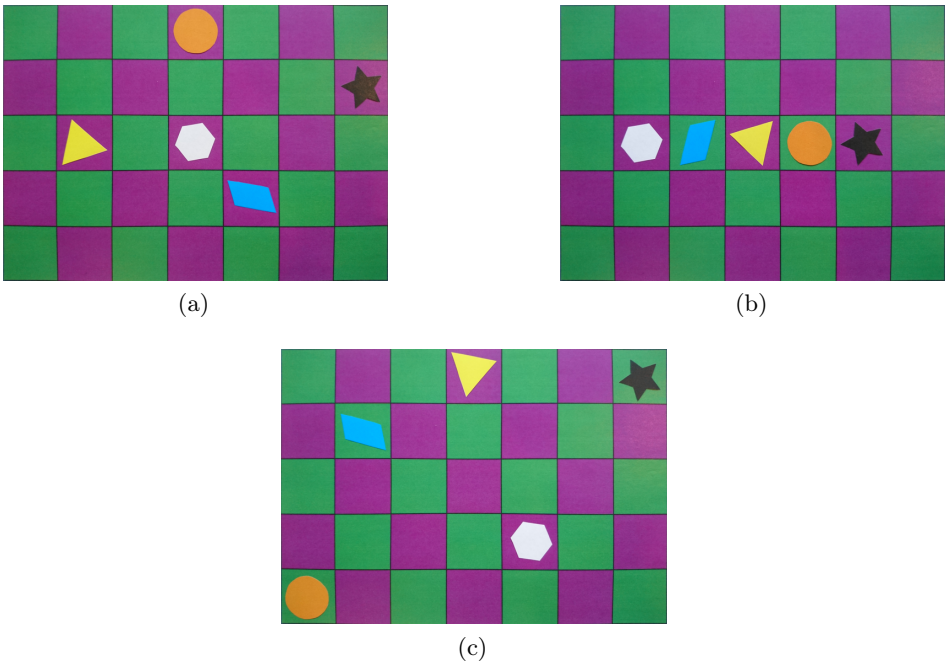
where $\boldsymbol{A}$ is the binary image where we want to find edges, $\boldsymbol{B}$ is a structuring element, and $\ominus$ is morphological erosion. The operation is commonly called *boundary extraction*. The shape of the extracted boundary depends on the structuring element we elect to use. For this task, a simple 3×3 structuring element of *all ones* will suffice.

a) **[0.5 points]** Implement a function that extracts the boundary from a binary image using arithmetic operations and morphological erosion as seen in Equation 1. Feel free to experiment with different structuring elements.

Apply your function on the noise-free binary image from the previous task and display the result in your report.

Task 5: Shape Recognition [3 points]

In this task, you will create a script / program that will be able to segment, locate, and record the shape type of all coloured shapes in a set of 2D colour images. Take a look at Figure 5 to see the three images we have made available to you. These are all attached with the assignment on *Blackboard*. An overview of all valid shapes can be seen in Figure 6.



(a) (b) (c)

Figure 5: Three images with a checkerboard background and various shapes. The orientation of each shape may vary.

This may seem like quite a daunting task due to how open-ended it is, thus to facilitate the process we have split the task into three subtasks. First, you will remove the grid making up the checkerboard, then you will isolate all shapes and label them as separate connected components, and finally, you will perform basic shape recognition.

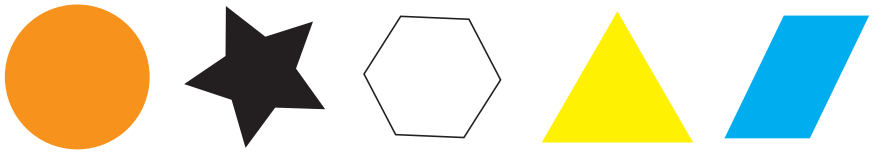


Figure 6: Overview of all valid shapes on the violet and green checkerboard. Notice that a black border has been added around the white hexagon for clarity.

Important: You may use any existing library / algorithm / function call to solve the subtasks. We only ask that you explain what you did to solve each subtask, as well as briefly explain how each technique you make use of works. For instance, if you want to use the Canny edge detector, then you would need to briefly explain the steps that make up the algorithm. Additionally, you *must* display the result of each subtask procedure for the three input images attached with the assignment.

Make sure to read / explore the documenation of whatever image processing / computer vision library you elect to use, for example, scikit-image in Python or various toolboxes in MATLAB.

a) **[0.8 points]** Implement a procedure that takes in one of the three supplied colour images and returns a binary image where most, if not all, of the checkerboard edges have been removed. By checkerboard edges we mean the edges spanning the image vertically and horizontally, thereby making up the checkerboard grid.

Below is a listing of a few algorithms and phrases that should guide you in the right direction:

- Canny edge detector
- Hough transform
- Mathematical morphology
- Sobel operator

Hint: Think about which colour space you work in at all times. For example, if you're having trouble with greyscale, you should try to work in some other space.

b) **[0.8 points]** Implement a procedure that takes in a binary image developed by the procedure in the first subtask. The procedure should clean up any remaining, problematic, noise and return a set of isolated shapes that can be analysed further. The result, could, for example, be an image where each shape is given its own label.

The labelling of shapes can, for instance, be achieved using connected-component labelling or the Moore neighbourhood algorithm.

c) **[1.4 points]** Implement a procedure that takes in a labelled image, or set, produced by the second subtask and returns a list of all the shape types appearing in the

image, along with their centroid. In other words, this procedure should recognise shapes and record their centre position.

Given a labelled shape within an image, there are several shape, or rather boundary, descriptors that you can easily compute in order to analyse them. Examples include centroid, diameter, perimeter length, area, eccentricity, and bounding box. Many of these, and more, are often computed using statistical moments[3].

For each of the three input images, visualise the detected shape centroids and shape types. Briefly discuss how well your shape recognition system works for the three images.

[3]Typically referred to as *image moments* in literature.

Optional: Morphological Skeletonisation [0 points]

This task is optional. The intention with this task is for you to gain further insights into the capabilities of mathematical morphology.

Skeletonisation, or medial axis transform, is a process for reducing an object to a set of points that have more than one closest point to the boundary of the object. That is, for a binary image, foreground pixels are removed until only a skeleton of the foreground regions remain. An example of a binary image along with its skeleton can be seen in Figure 7.

(a) (b)

Figure 7: (b) is the skeleton of the box in (a).

One way to find the skeleton(s) of a binary image is to iteratively apply morphological thinning until there is no change in the processed image. This process can be seen outlined in pseudocode below:

Algorithm 1: Morphological skeletonisation.

Input : Binary image Image $\boldsymbol{B}$
Output : Skeleton image **S**
$\boldsymbol{S} \leftarrow \boldsymbol{B}$
do
  $\boldsymbol{S}_{previous} \leftarrow \boldsymbol{S}$
  $\boldsymbol{S} \leftarrow$ perform morphological thinning on $\boldsymbol{S}$
while $\boldsymbol{S} \neq \boldsymbol{S}_{previous}$;

Now that we know how to skeletonise a binary image, we can focus our attention on how to *thin* a binary image. There are multiple ways to do this, but we will be using the hit-or-miss transformation[4]. This is a morphological operation that takes in a binary image and *two* structuring elements collectively called a composite structuring element. The hit-or-miss transformation $\otimes$ consists of two applications of erosion: one between the binary image and the first structuring element, and a second between the complement of the binary image and the second structuring element. For a binary image $\boldsymbol{A}$ and structuring elements $\boldsymbol{B} = (\boldsymbol{C}, \boldsymbol{D})$, the hit-or-miss transformation is defined as:

$$\boldsymbol{A} \otimes \boldsymbol{B} = (\boldsymbol{A} \ominus \boldsymbol{C}) \cap (\boldsymbol{A}^c \ominus \boldsymbol{D}) \tag{2}$$

[4]Sometimes also referred to as the hit-*and*-miss transformation.

Intuitively, the currently inspected pixel will only be set to the foreground colour if: (i) $\boldsymbol{C}$ translated over the pixel *hits*, and (ii) $\boldsymbol{D}$ translated over the pixel *misses*.

Using the hit-or-miss transformation, a single morphological thinning step can be performed by running the following operation *eight* times:

$$\boldsymbol{X}_i = \boldsymbol{X}_{i-1} \cap (\boldsymbol{X}_{i-1} \otimes \boldsymbol{B}^{(k)})^c \qquad \text{for} \qquad k = 0 \ldots 8 - 1 \tag{3}$$

where $\boldsymbol{X}_0$ is the input binary image and $\boldsymbol{B}^{(k)}$ is the kth composite structuring element. For practical purposes the complement $(\cdot)^c$ can be computed using a logical NOT, while the intersection $\cap$ can be computed using a logical AND.

As mentioned above we need to apply Equation 3 eight times, with a different composite structuring element each time. These consist of two groups of which are unique, while the rest are simply rotations. The two unique composite structuring elements can be seen in Figure 8.

$$\boldsymbol{C}^{(0)} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \boldsymbol{D}^{(0)} = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

(a)

$$\boldsymbol{C}^{(1)} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 1 & 1 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \qquad \boldsymbol{D}^{(1)} = \begin{array}{|c|c|c|}\hline 0 & 1 & 1 \\ \hline 0 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

(b)

Figure 8: The two structuring elements on the left-hand side correspond to the first composite structuring element $\boldsymbol{B}^{(0)}$, while the two on the right correspond to the second composite structuring element $\boldsymbol{B}^{(1)}$. The centre pixel is the reference pixel for all structuring elements.

By considering all 90° rotations of the *two* composite structuring elements in Figure 8, we get a total of eight composite structuring elements. All of these have to be used according to Equation 3, in order to apply a *single* thinning operation. With this knowledge you should have no problem implementing skeletonisation for binary images.