# Assignment 4

## General Info

- The work can be done anywhere where MATLAB and the related toolboxes are available.
- A **written report** is required. The report is free form but should include results, figures, and any code you wrote, as well as discussions of what you observe.
- Reports should be submitted as an electronic copy in **PDF** format on **Canvas** before the due date.
- Late submissions will not be accepted. If you have not completed all the problems, submit what you have.
- Better documentation and clear but concise discussions of your work improve our ability to fairly mark your report. Make sure your report is well structured and organized. Your report has to follow the order of the questions. Give all relevant code right before the results and discussion of each part, not in an appendix.
- This assignment has 5 questions.

## 1. Hough Transform

The original Hough transform was designed to detect straight lines. A big advantage of this approach is robustness of the segmentation results, given its relative insensitivity to occluded or noisy data.

In this problem, you will learn about the Hough transform. Using the provided code, you will first explore line detection and use that knowledge to modify the code such as to enable circle detection in images.

**Getting started:** To run the provided Hough transform code:

- Download the files provided within the assignment.
- Open the folder *"\matlab_student_HW4\Q1\"*
- Run '`CVproj`'.
- TIP: If the program window is not within the screen boundaries, this can be fixed by increasing the screen resolution, or by:
  - In windows, press the keys Alt-spacebar then press the key 'M' to move/drag the window with the keyboard arrows, or 'X' to maximize it.
  - In Linux, press and hold the Alt key while dragging the window with the mouse.

**Brief explanation of the buttons in the provided Hough Transform Program**

- *Choose Image*: Selects an image (JPEG, TIFF, BMP, PNG, HDF, PCX, XWD, and PGM).
- *Detect edge*: Applies edge detection to a selected image using different gradient kernels (Sobel, Prewitt, Roberts), sub-pixel resolution, or other methods such as: Canny or looking for zero crossings after filtering the image with a Laplacian of Gaussian filter.
- *Hough*: Performs Hough transform on the detected edges. You can specify the intended resolution for the resulting vote histogram.
- *Unhough*: Extracts plausible lines from the vote histogram matrix. You can specify a vote threshold value that will effectively control the number of selected lines.

Explore the GUI by choosing the sample image ‘`image.bmp`’. Concisely explain how the program works on the image by specifying what each of the files starting with `CV*.m` do.

a) Check what MATLAB prints as results when ‘`unhough`’ is pressed. Explain the results.

b) Explain what do the ‘`Threshold`’, ‘`Number of distances`’, ‘`Number of angles`’, and ‘`Vote threshold fraction`’ in the program window refer to.

c) Generate or find a new image of your choice with suitable structures to be processed by the hough program and run the procedure on your image. Tune the parameters to give the results you desire.

d) Write your own MATLAB code to extend the Hough transform to circle detection and test your new code on the images ‘`circle.bmp`’ and ‘`circle_occluded.bmp`’.

e) Make/find a new image of your choice with suitable structures to run your code on for further testing.

## 2. Texture Analysis

The Fourier spectrum is well suited for describing global texture patterns in images which manifest themselves as high-energy bursts in the Fourier domain. In this question, you will learn how to use spectral information as descriptors for images with periodic patterns.

a) Load *‘periodic.jpg’* and calculate the Fourier transform of the image, then center the spectrum such that DC is in the middle of the Fourier image.

b) Using the magnitude spectrum, from the DC center, select a radius of 20 samples. Now with this fixed radius, find all the magnitude of the Fourier transform as a function of theta, $S(\theta)$, where $\theta$ is from 0 to 360 degrees with a step of 10 degrees. Plot $S(\theta)$ vs $\theta$ and discuss how this can be used as a textural descriptor for this image.

c) Similarly to part(b), instead of varying the angle $\theta$, select a fixed angle $\theta$ of 45 degrees and vary the radius ‘r’ starting from 0 to the maximum radius in the frequency spectrum. For every value of ‘r’, find the corresponding magnitude of the Fourier transform as a function of radius $S(r)$. Plot $S(r)$ vs ‘r’ and discuss.

d) Repeat part (b) and (c) using *‘grid.jpg’*. Comment on your findings and compare to your earlier findings in (b).

## 3. Morphological Operations

In this question, you will explore morphological operations and their use for preprocessing images and analyzing objects present within them.

a) Load MATLAB's image 'rice.png'. Try to segment the rice grains using Otsu's thresholding (*graythresh* function in MATLAB). Explain the issues, if any, related to the background illumination and its effects on your rice grains segmentation results.

b) Use morphological opening to remove the rice grains from the image by selecting a disk structuring element with radius 15.

c) Subtract the image extracted in (b) from the original "rice.png" and comment on the resulting image.

d) Use *'imadjust'* to enhance the contrast of the image obtained in (c).

e) Repeat your segmentation by thresholding the preprocessed rice image obtained in (d) and comment on the results in comparison to those obtained in (a).

f) Suggest additional morphological operations to remove any remaining noise components, such that your resulting segmented binary image of the rice grains is as accurate as possible.

g) Using your binary image in (f), use MATLAB's *'bwconncomp'* function to count the number of rice grains present in the image. There are 100 rice grains present in the image, does you result match with the correct number? Discuss why/why not.

h) Use *'regionprops'* to calculate the area of each of the connected grains and show the maximum and minimum area of the grains present from the connected component analysis. Why there is such a variation? What do you understand from the numbers? Use histogram to show the different areas calculated for the rice grains.

## 4. Identifying geometrical shapes from images

Here you will learn how to detect different geometrical shapes in an image with the use of shape descriptors.

a) Run Q4_student.m and explain the operations performed in the code as well as the output results.

b) Change the threshold explaining what it controls and investigate if the rubber bands can also be labeled as circles.

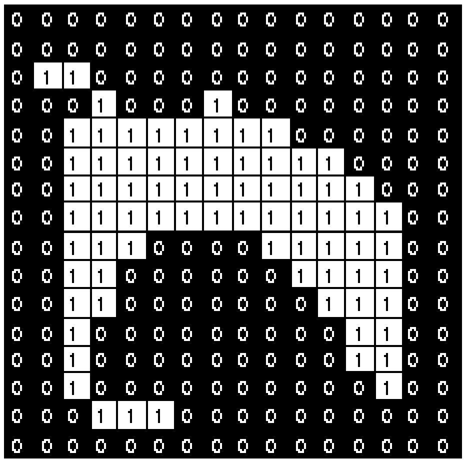
c) Implement another descriptor in the code to enable detection of rectangular shapes.

d) Use 'rectangle_paintings.png' as input in the modified code to find all the rectangles present in the image.

## 5. Skeletonization of Objects

Skeletons constitute one of the fundamental representations of objects. A skeleton simplifies the object shape using a recursive thinning operation. In this problem, you will implement the skeletonization algorithm discussed in class.

a) Load the provided **“input.mat”** file to load the object with the structure below and then zero pad around the edges.
.



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

b) Use the following structural masks to thin the image generated in (a). In this method, the object is sequentially thinned from all 4 directions. For every direction, there are three different masks as shown in the image below. Implement a function that thins the image from the west, then north, then east and then south (perform **one iteration of each only** in the described order). Remember to shrink the pixels only after one direction of attack has been completed. The attacks must be performed sequentially starting from west, north, east and then ending with south.

c) Repeat step (b) iteratively until no further change occurs to the final thinning result. Show the final output which constitutes the object’s skeleton.

d) Generate a binary rectangle with perturbed sides, where pixels are sticking a bit. Use the same code on this newly generated image and see how the result looks like. Could the algorithm thin the shape correctly?

**<u>End of assignment 4</u>**