CS484/555 - Introduction to Computer Vision
Homework 2
Report

Efe Tarhan
22002840
Electrical and Electronics Engineering

Content

I) Introduction

In this project, the main task is to match colored images of rotated books with the original versions and estimating the rotation amount in terms of degrees by utilizing Canny edge detector and Hough transform. Purpose of this homework is to understand the pipeline of an image matching algorithm while using several classical computer vision techniques end-to-end to obtain a result. The algorithm will operate by creating histograms of the number of lines belonging to the specific angle ranges for their slope. There are also multiple hyperparameters of the algorithms that must be adjusted for this task to get optimized and results with good prediction and estimation capacity. One of the books in the sample set and its rotated version is shown in the Figure 1.

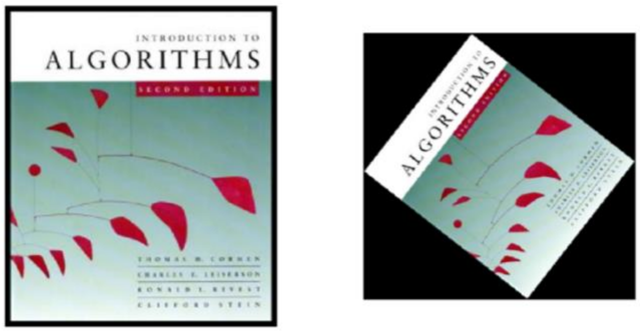

Fig.1 First image in the homework images with its rotated version

The images will be processed using the built-in functions of MATLAB. Two of the main algorithms used for this project are the Canny edge detector and Hough transform. The theory of these methods will be briefly described in this section while details of implementation will be given in the following chapters. After reading the images using MATLAB the next task will be to detect the edges and lines in the image so that they can be used as features of the given image. For this task, the Canny Edge Detector is used.

Canny Edge Detector

This technique is being used for detecting and filtering edges from images as thin straight lines using techniques like **non-maxima suppression** and **hysteresis thresholding**. These two techniques cause this algorithm to become a strong alternative compared to other classical edge detection techniques. Initial step of canny edge detector is the usage of gradient operators to obtain image gradients in the x and y directions. Magnitude of these gradients are obtained by using the following formulas.

$$|\nabla| = \sqrt{\nabla_x^2 + \nabla_y^2}$$

$$\angle(\nabla) = \arctan\frac{\nabla_y}{\nabla_x}$$

Following this calculation, the gradients that are equal to or smaller than the neighbors along the gradient directions are equated to zero which is called as the non-maxima suppression step of the edge detector. This operation is illustrated in Figure 2.

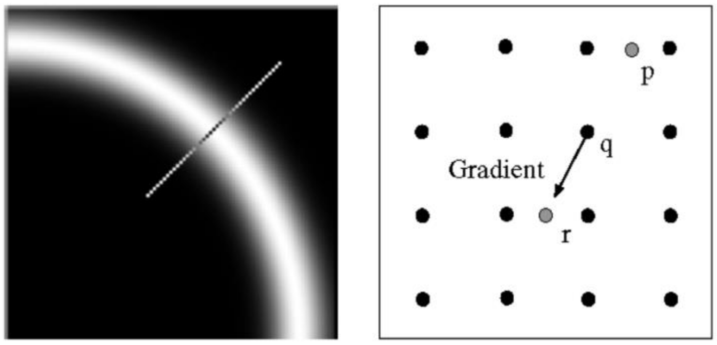

Fig.2 Illustration of non-maxima suppression algorithm

Finally, a hysteresis thresholding technique is applied to the image obtained after the non-maxima suppression algorithm. Hysteresis thresholding means finding edges with high amplitude values and then applying a lower threshold to the neighboring pixels if they follow the edge to obtain a binary image. The illustration of the technique can be seen from Figure 3.

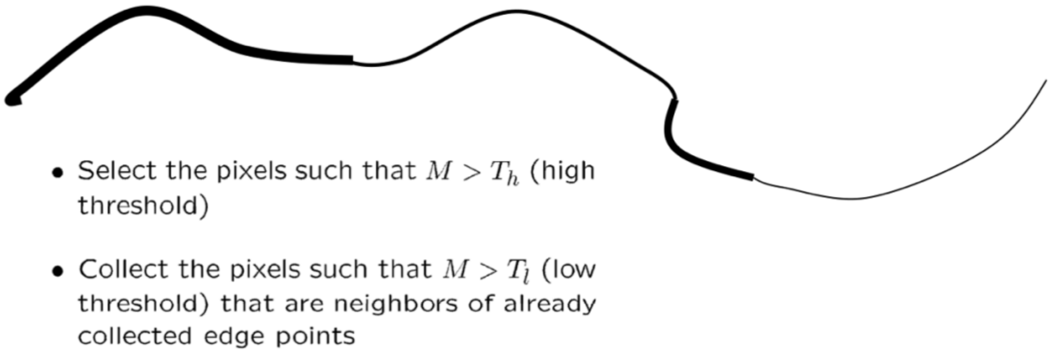
• Select the pixels such that $M > T_h$ (high threshold)
• Collect the pixels such that $M > T_l$ (low threshold) that are neighbors of already collected edge points

Fig.3 Illustration of hysteresis thresholding algorithm

After implementing the given two algorithms the canny detection is completed for the given images. There are 3 important hyperparameters of the Canny edge detector implementation of MATLAB [1]. The high threshold (T_h), low threshold (T_l) and sigma which is the standard deviation of the Gaussian filter applied to the image before the thresholding application.

Hough Transform

Hough transform is used for detecting edges, lines and specific parametric curves from an image using transformation of the pixel to the space of the parameter space of the investigated shape. Each pixel in the image is transformed into a sinusoidal curve in the parameter space of lines because a line can be represented with the following formula:

$$\rho = x sin(\theta) + y cos(\theta)$$

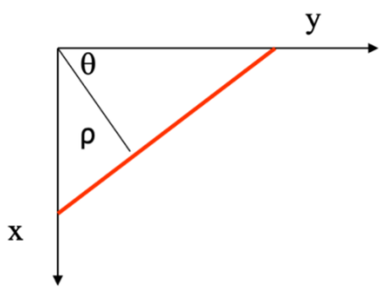

Fig.4 Representation of a line in the image using the parameters θ and ρ

Therefore, there is a set of parameters θ and ρ such that the point (x, y) satisfies. These set of parameters constitute a sinusoidal function where a line can be detected by the points that share the same θ and ρ at different crossing points of the sinusoidals. There are not any hyperparameters of the Hough transform implementation of MATLAB [2]. The function takes a greyscale image as input and gives the Hough transform matrix as the output. This matrix is latter used as an input to the MATLAB function “houghpeaks” where it selects the proper peaks and parameters that can describe lines in the image [3]. **Maximum number of peaks** parameter determines the maximum number of peaks to select in the transformed space whereas the **threshold** parameter determines the minimum number of crossings for that point to be considered as a candidate peak. The “houghpeaks” function outputs a struct object called peaks where each peak element has specific attributes. MATLAB has also an additional function for detecting the finalized lines called “houghlines” where it takes the variables the transform matrix, ρ values, θ values and peaks structs as input and gives the lines in the image as output [4]. The “houghlines” function takes 2 main hyperparameters which are the **MinLength** and **FillGap** parameters. The “MinLength” determines the minimum length as pixels that makes them a line whereas the “FillGap” parameter determines the number of pixels between two lines that puts them into the same Hough bin or not.

After extracting the lines from the images, histogram for each image is built using the slopes of the lines and length of the lines as lengths. After creating the histogram of all images, the histograms of the rotated images are compared with the template ones circularly shifting the bin edges to both detect the image and their rotation amount.

II) Canny Edge Detector

After reading the images using the “imread” function of MATLAB, the images are saved as 3 dimensional tensors where they are converted to greyscale using “rgb2gray” function of MATLAB. After that the canny edge detector has been used with different parameter values. Effect of each of these parameters described in the introduction part will be described below.

As explained in the introduction part, the high threshold parameter (T_H) determines the initial points to select on the magnitude of the gradient of the image after applying the non-maxima suppression algorithm. Effect of changing this parameter while fixing other Canny detector parameters is shown in Figure 5.

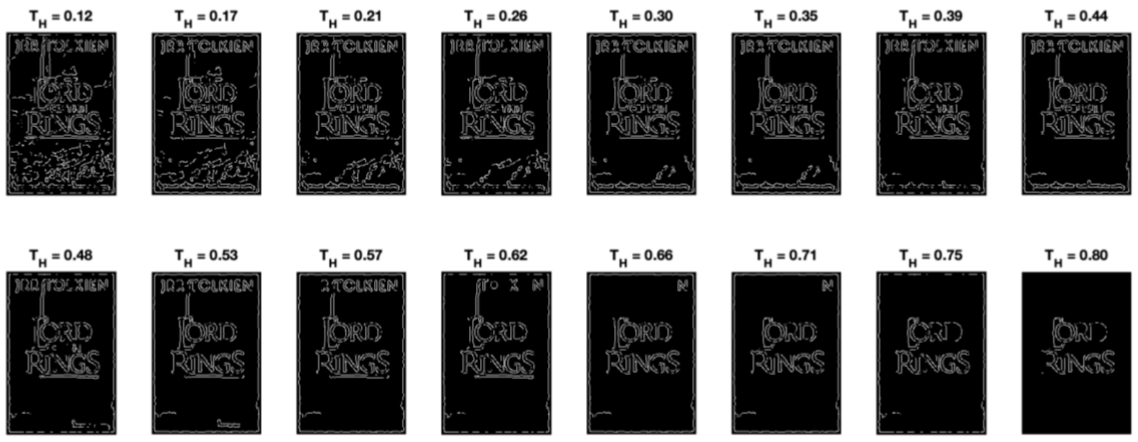

Fig.5 The effect of low threshold T_H of Canny edge detector w/ $\sigma = 2, T_L = 0.1$

Increasing the T_H to a level above the threshold 0.4 kills some of the necessary details of the image while keeping it lower than 0.2 preservers the gradient magnitude contours of unnecessary details of the image which can be considered as the noise of the system for this task. To only keep the relevant curves and lines to the task the lower threshold parameter has been selected as 0.2 so that it preserves all details in the image while suppressing unnecessary details of the image.

Selecting the low threshold (T_L) is the next step for the hysteresis thresholding where the lower threshold represents the threshold for keeping the pixels after selecting the pixels that surpass the higher threshold of the hysteresis thresholding. The effect of range of T_L values while fixing other parameters are shown in Figure 6.

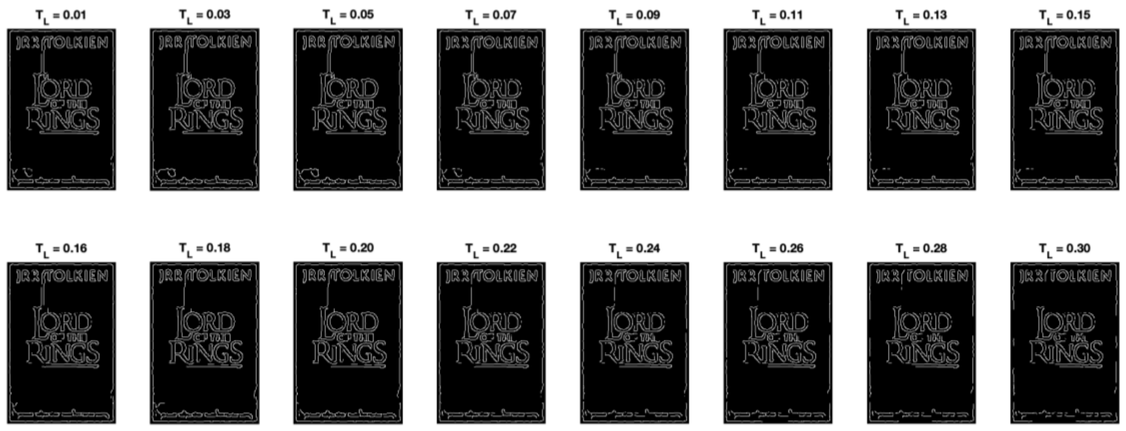

Fig.6 The effect of low threshold T_L of Canny edge detector w/ $\sigma = 2, T_H = 0.4$

Another parameter of the canny edge detector is the sigma value which is the standard deviation of the Gaussian blur filter. With the increasing sigma value, the images get blurrier and details including noises of the image that are desired to be left out are eliminated. Although eliminating details to a degree is necessary increasing the sigma value eliminates too many details that may lead important information to be lost. Effect of changing the standard deviation of the Gaussian filter is shown in Figure 7 while other parameters are fixed.

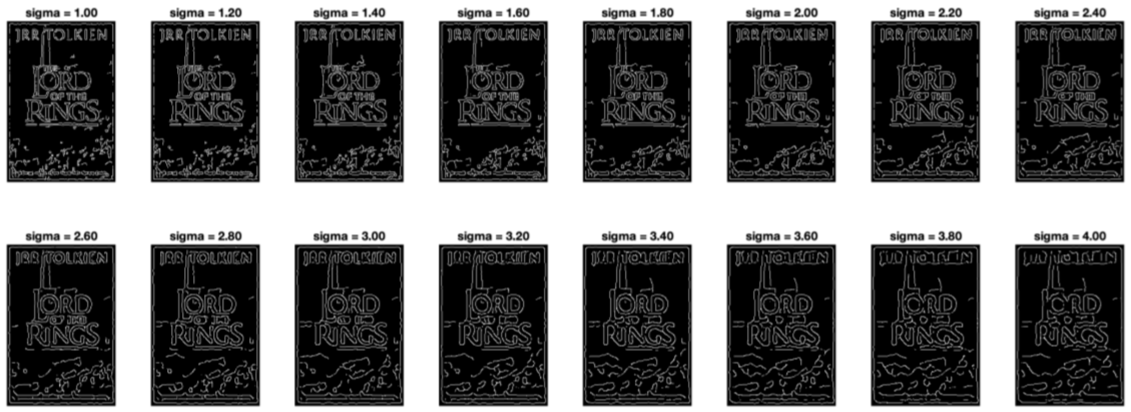

Fig.7 The effect of σ of Canny edge detector w/ $T_L = 0.1, T_H = 0.2$.

After fixing the parameters of the Canny edge detector as $T_L = 0.1, T_H = 0.2$ and $\sigma = 2$, edges of all books are extracted using these hyperparameters which can be seen in Figure 8 and 9.

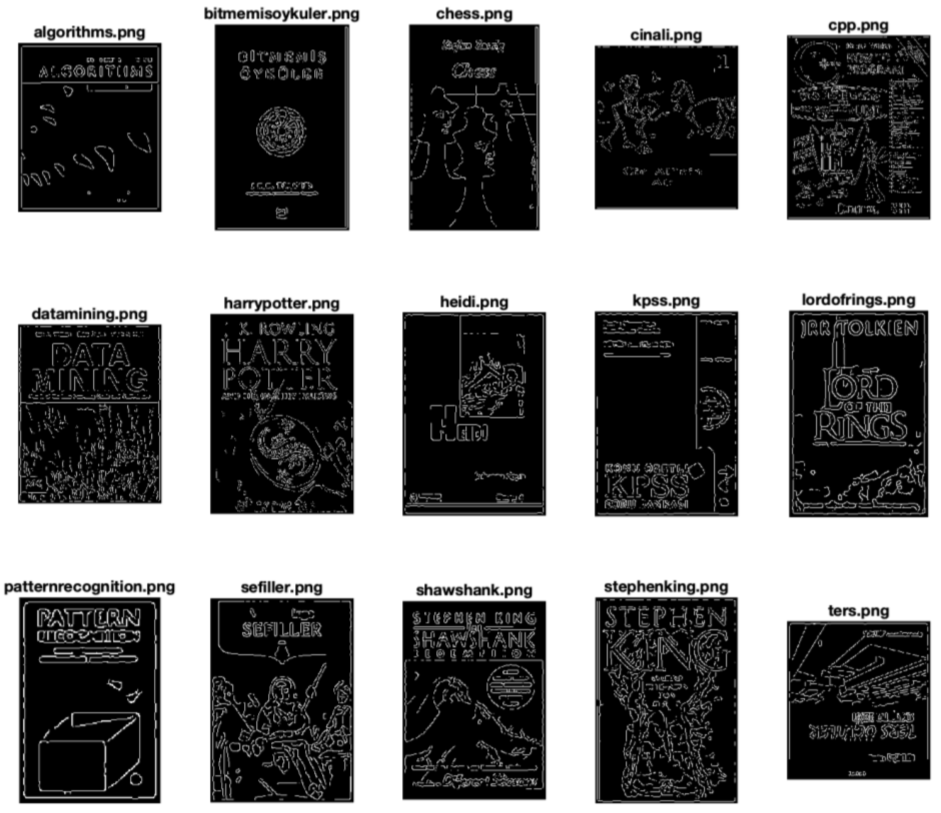

Fig.8 Edge detected versions of template images w/ $T_L = 0.1, T_H = 0.2, \sigma = 2$.

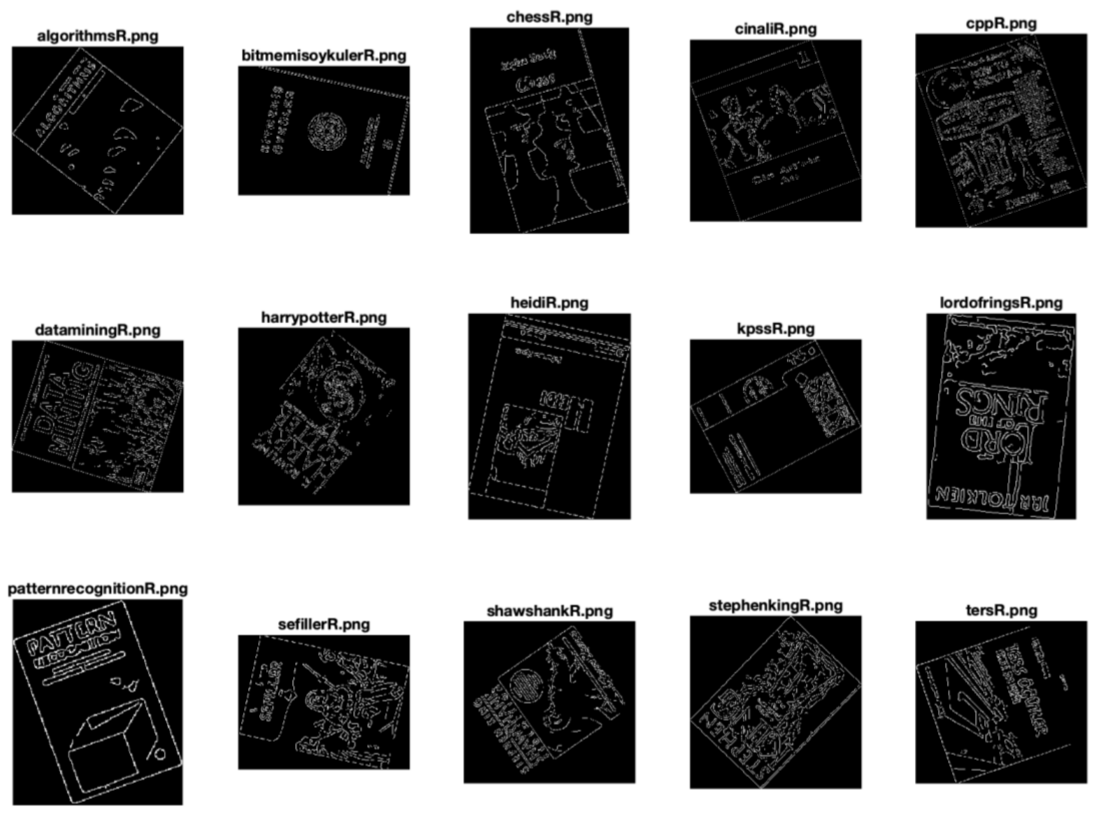

Fig.9 Edge detected versions of rotated images w/ $T_L = 0.1, T_H = 0.2, \sigma = 2$.

The resultant edge-extracted images will be used for Hough transform and lines along the edges will be selected and binned for histogram creation.

III) Hough Transform

After obtaining the edges of the image in binary format they are used with the Hough transform algorithm to extract the line segments in the image with their Hough parameters. There are 4 important hyperparameters of Hough transform which will be tested in this section. The initial parameter to be searched for is the maximum number of allowed peaks in the Hough matrix which determines the maximum number of lines that can be extracted from an image. The effect of changing the number of maximum peaks while keeping other hyperparameters fixed can be seen in Figure 10.

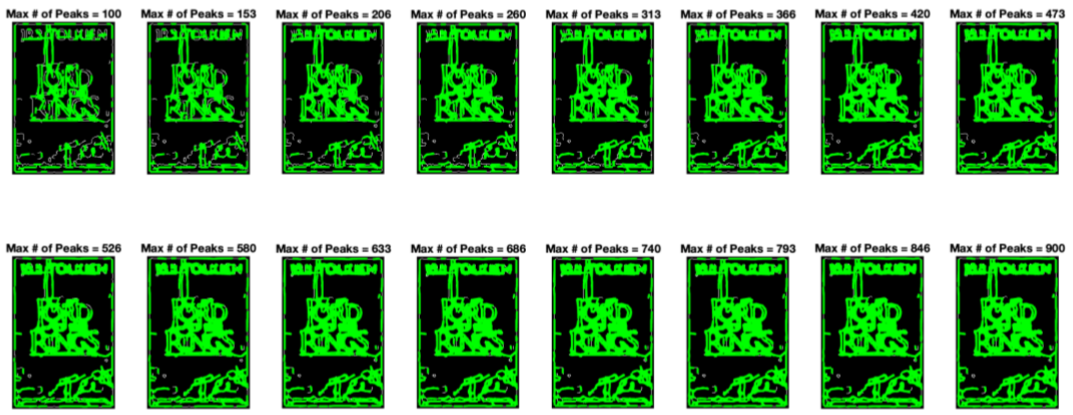

Fig.10 Effect of number of maximum peaks parameter on the detected lines

As it can be seen allowing number of peaks smaller than 300 affects the number of detected important lines including lines that contain important information about the orientation of the image. Also increasing the maximum number of peaks too much creates many lines that some of them might not be significant for the image and therefore will create a problem while comparing images with their rotated versions. Therefore, the maximum number of peaks parameter is selected as 600 to satisfy both conditions.

The peak ratio parameter determines the threshold for selecting a peak as a peak where it consists of crossings of sinusoidals. The threshold parameter of the "**houghpeaks**" function is determined in term of a percentage of the peak. Effect of peak ratio parameter on one of the images can be seen from Figure 11.

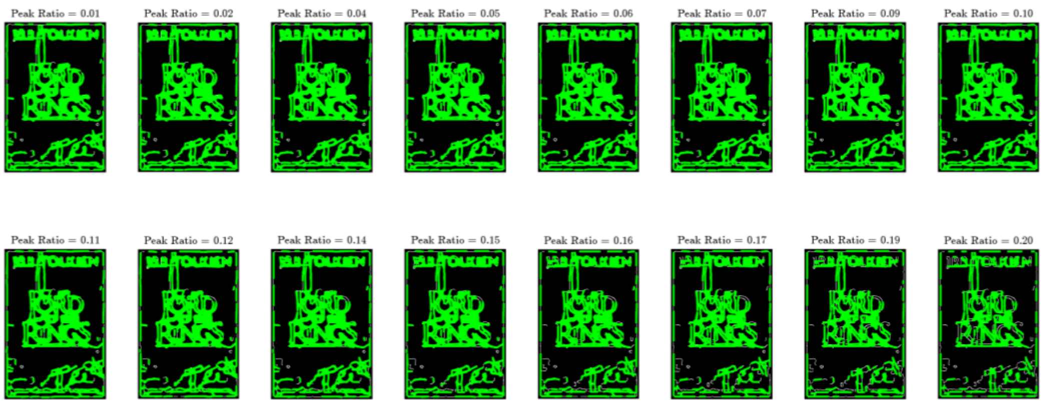

Fig.11 Effect of peak ratio parameter on the detected lines

As it can be seen from the image keeping the threshold too low causes many of the lines to be picked whereas keeping it too low causes some of the important lines to be missed. One of the optimal values that result with high detection accuracy for this parameter is preferred which is 0.02.

After obtaining the parameters for the peaks of the Hough transform, lines can be extracted using the Hough matrix and “houghlines” function of MATLAB. “FillGap” is one of the important parameters of this function. It determines the which distance is acceptable between two points for them to be considered as lines. This parameter is especially important for non-horizontal or non-vertical lines since they do not appear as continuous lines on the image, rather dotted line pieces. Effect of trying different parameters for **FillGap** while fixing other parameters is shown in Figure 12.

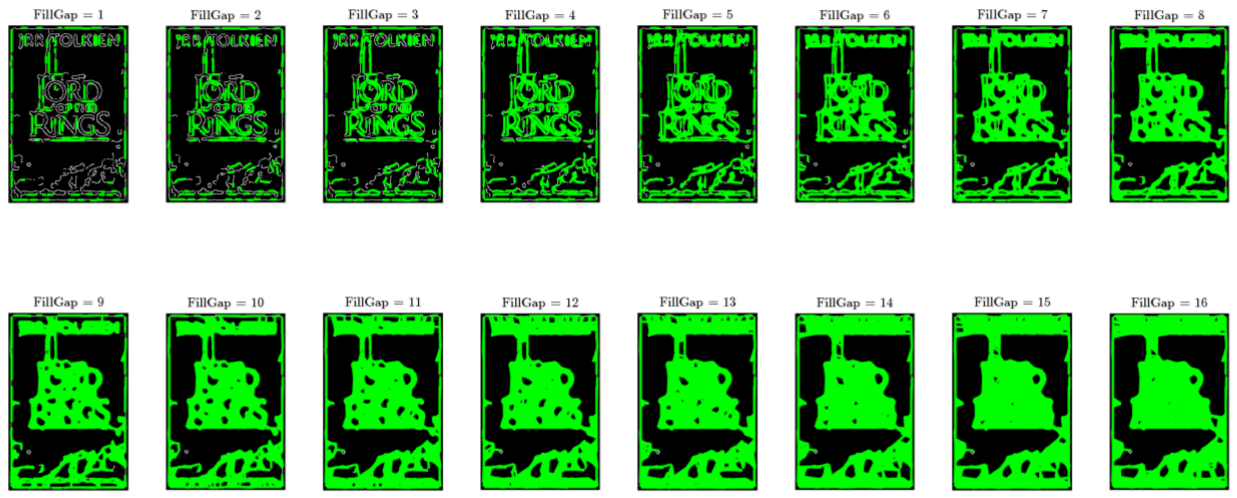

Fig.12 Effect of the **FillGap** parameter on the detected lines

As it can be seen increasing the maximum number of distances to create a line gives permission to many peaks in the image to become line pairs where it results in the saturation of number of lines. For example, if the FillGap variable is set larger than 10, the original lines that belong to edges become less visible and the system captures unnecessary lines that might cause uniformity and hence loss of variance among the data. Therefore, the FillGap variable is selected in an optimal manner as 6 where the maximum number of lines are captured while unnecessary ones are left out.

Another hyperparameter of the system is the minimum length of the lines for them to be accepted as lines that belong the image. Effect of changing the MinLength variable is shown in Figure 13 where other variables of the Hough transform related function are kept fixed.

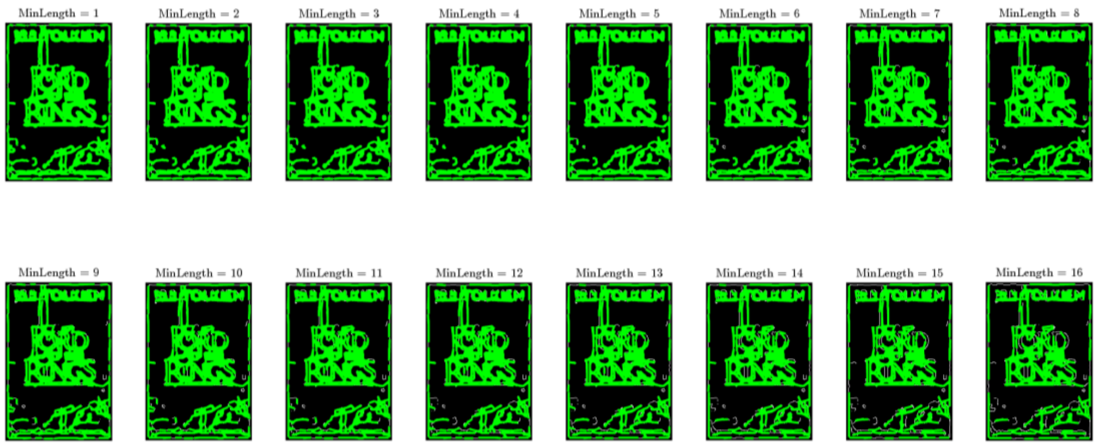

Fig.13 Effect of the **MinLength** parameter on the detected lines

Similar to the FillGap variable decreasing the MinLength parameter allows unnecessary lines to be kept by the algorithm. Again, an optimal choice has been made as 8 where the number of detected peaks is maximum for the given variable.

After fixing the parameters of the Hough Transform with maximum number of peaks equal to 400, peak ratio equal to 0.02, FillGap equal to 6 and MinLength equal to 8 all images are transformed, and their lines are extracted using these parameters and related functions.

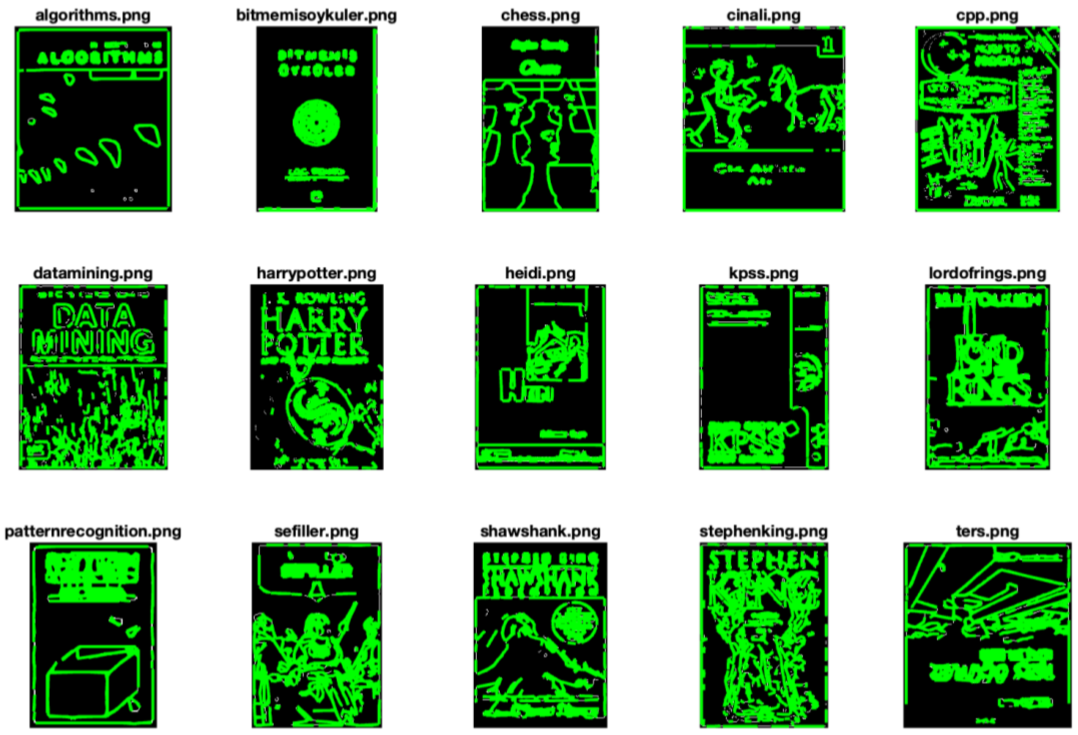

Fig.14 Detected lines of the template images.

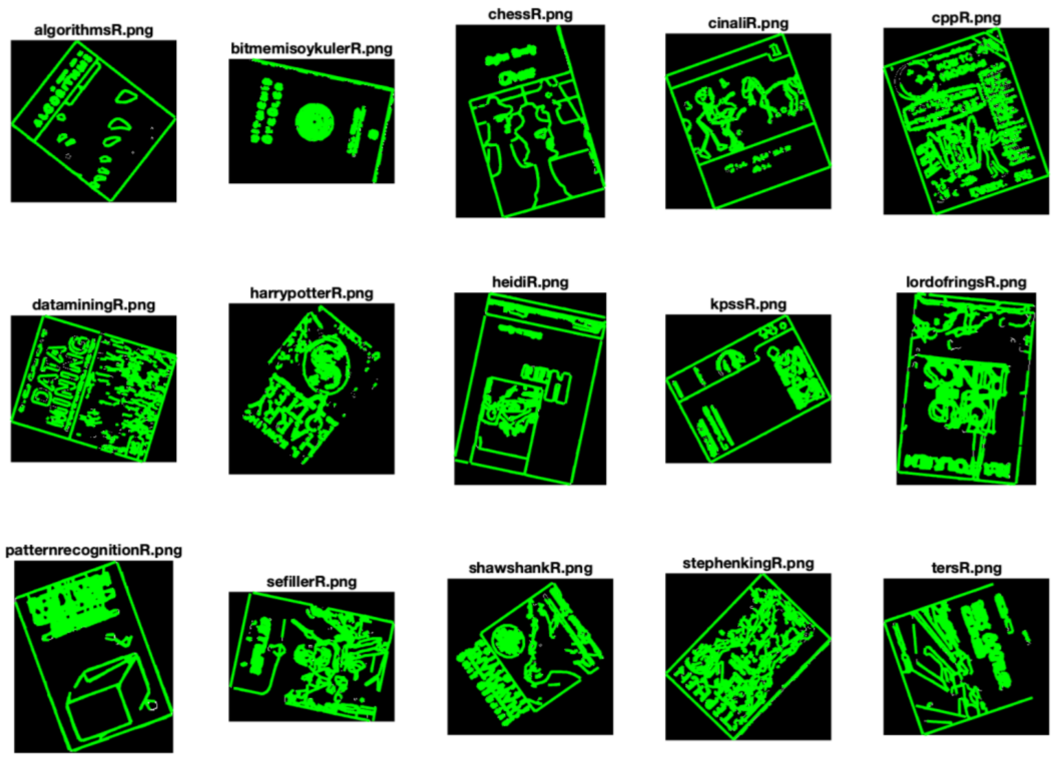

Fig.15 Detected lines of the rotated images.

A sample zoomed image with beginning and ending of the lines that are denoted can be seen from the Figure 16.

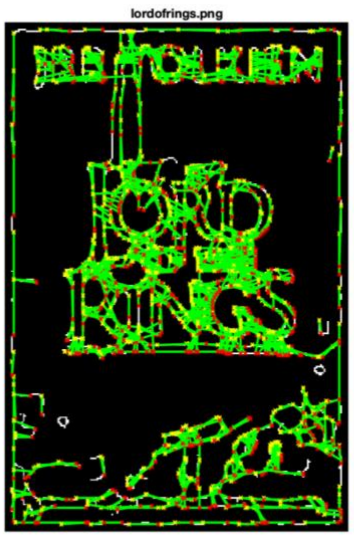

Fig.16 lordofrings.png image after Canny edge detection and Hough transform

Following the Canny edge detection and Hough Transform, next step is to create histograms with the slopes of the lines while weighting them with their lengths.

IV) Histogram Encoding

After extracting the lines histograms are created by dividing the angle range between $0° - 180°$ to a given number of bins. Following that the lines are divided into these bins being weighted by their length. Since the Hough transform can only detect rotations between $-90°$ to $90°$ degrees. Rotations larger than $180°$ will be ambiguous and can be confused with same rotation symmetric with the origin. In Figures 17 and 18, histograms of the book lordofrings.png and lordofringsR.png are displayed respectively by changing the parameter of bin counts.

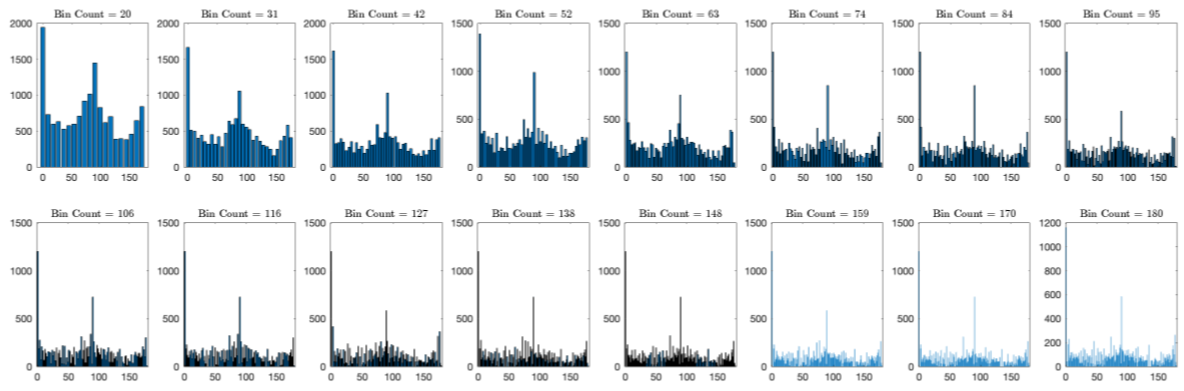

Fig.17 Histogram of lordofrings.png with for different number of bins

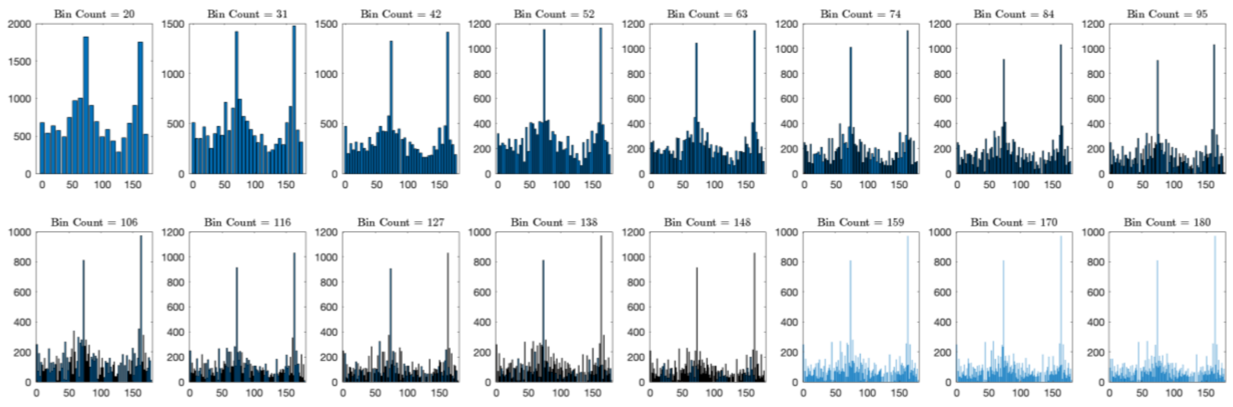

Fig.18 Histogram of lordofringsR.png with for different number of bins

It can be seen from the images that increasing the number of bins increases the detail and resolution of the angles but while doing that it strengthens the peeks at bins corresponding to 0 and 90 degrees which can result with a system that causes the data to not easily be separated from each other since all images mostly contain peaks around those angles. To avoid this problem an optimal number of bins must be selected. By some trials and observing the effect of changing the number of bins on the histograms, 60 bins are selected as a value for obtaining interpretable solutions where the algorithm gives a $3°$ degrees of resolution. Histograms of all template and rotated books can be seen from Figure 19 and 20 respectively.

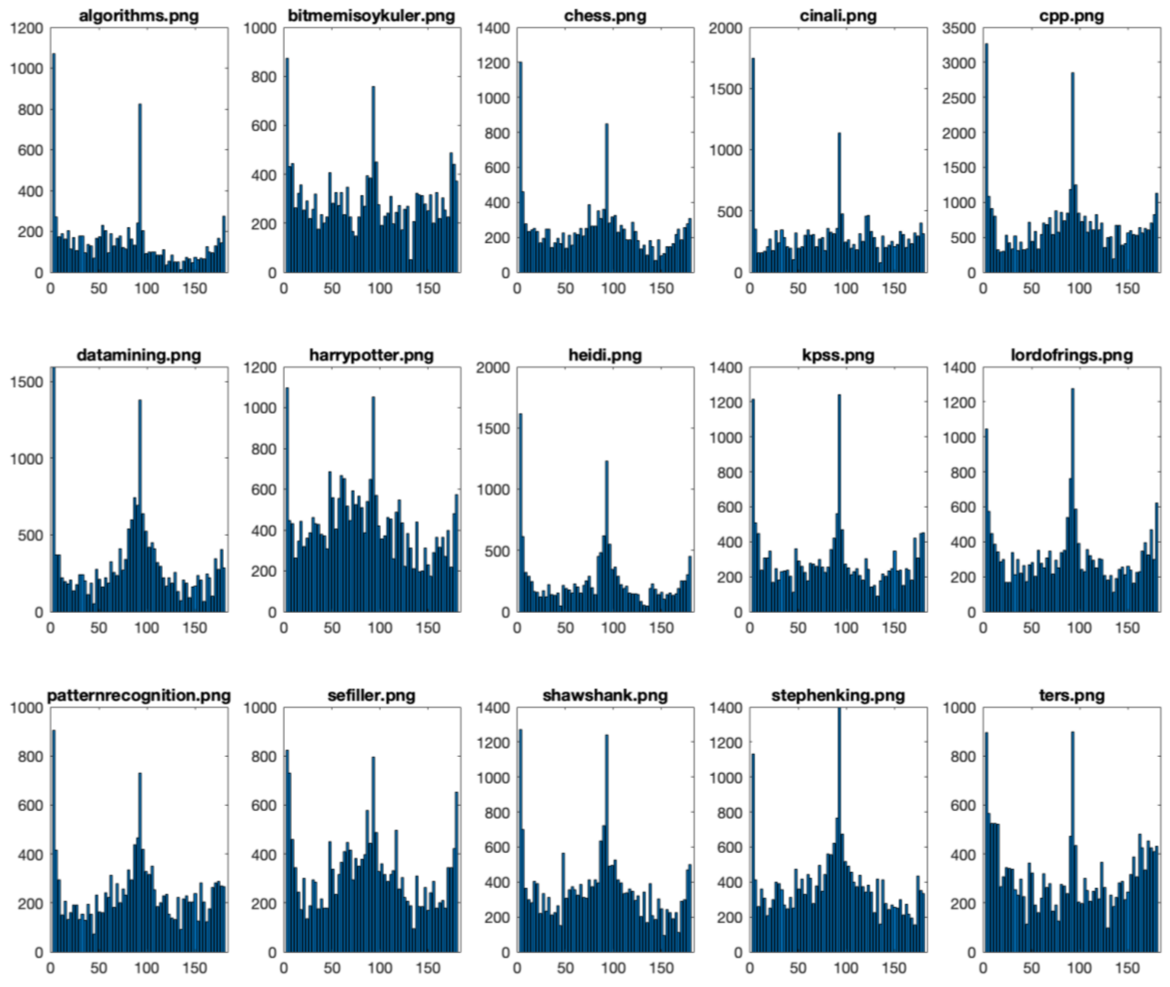

Fig.18 Histograms of template images constructed using defined parameters.

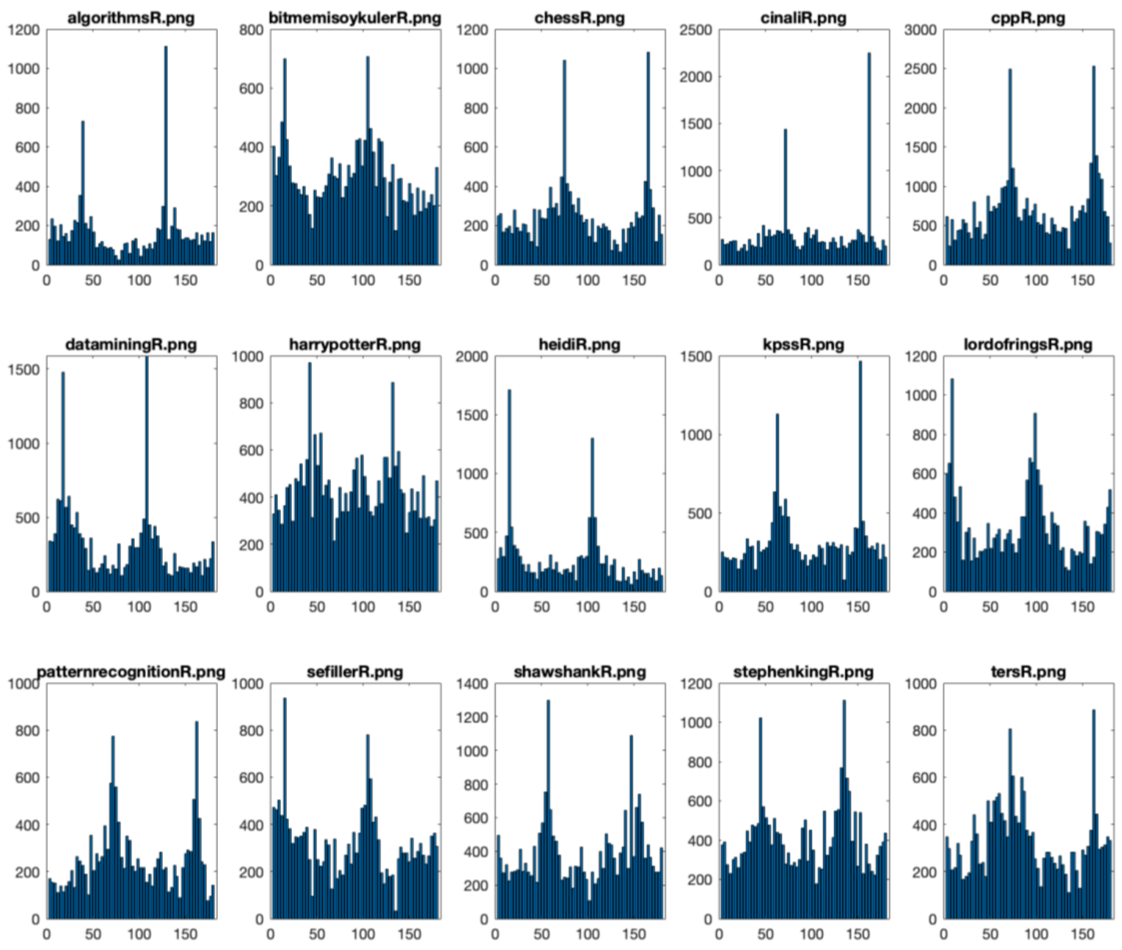

Fig.19 Histograms of rotated images constructed using defined parameters.

After generating the histograms with the final parameter values given in Table 1. The histograms will be compared with each other to find the matches.

Parameters	**Values**
T_L	0.1
T_H	0.2
σ	2
Max # of peaks	600
Peak Ratio	0.02
FillGap	6
MinLength	8
Bin Count	60

Table 1. Table of all parameters used throughout the detection pipeline.

V) Book Matching and Rotation Estimation

In this part the comparison of the histograms is done by using circular shifts and Euclidian distance. The histograms are saved as MATLAB cell objects where each instance of the shell corresponds to a histogram. Histograms are therefore column vectors with dimension equal to number of bins. From below the used algorithm to compare each class can be seen in Algorithm 1:

```
templatehist <- $H_T(i) \in R^{\#\ of\ bins}$  $\forall i \in 1,2,3 \ldots 15$

rotatehist <- $H_R(i) \in R^{\#\ of\ bins}$  $\forall i \in 1,2,3 \ldots 15$

matched <- $M(i) \in RxR$  $\forall i \in 1,2,3 \ldots 15$

for i in rotatehist:
   min_distance = inf
   min_rotation = 0
   for j in templatehist:
      for q = 1:bin count-1:
            new_template = rotate_circular_right(template_hist,q)
            dist = (new_template-i)^2
            if dist < min_distance:
                  min_distance = dist
                  min_rotation = q
                  min_book = j

   matched(i) = (min_book,min_rotation)
```

Alg.1 Matching algorithm for histograms

By using this matching algorithm, the books and their corresponding rotation angles are selected such that the Euclidian distance is minimized by sweeping over the template images and rotation angles which is number of circular shifts of the histogram. Results of applying this algorithm is demonstrated in the Table 2, where all images are detected correctly, and their rotation angles are also close or equal to the actual rotation amounts of the image.

Rotated Book Name	**Matched Template Name**	**Rotation Angle in Degrees**
algorithmsR.png	algorithms.png	$54° - 57°$
bitmemisoykulerR.png	bitmemisoykuler.png	$78° - 81°$
chessR.png	chess.png	$18° - 21°$
cinaliR.png	cinali.png	$21° - 24°$
cppR.png	cpp.png	$21° - 24°$
dataminingR.png	datamining.png	$75° - 78°$
harrypotterR.png	harrypotter.png	$141° - 144°$
heidiR.png	heidi.png	$168° - 171°$
kpssR.png	kpss.png	$120° - 123°$
lordofringsR.png	lordofrings.png	$174° - 177°$
patternrecognitionR.png	patternrecognition.png	$21° - 24°$
sefillerR.png	sefiller.png	$78° - 81°$
shawshankR.png	shawshank.png	$126° - 129°$
stephenkingR.png	stephenking.png	$138° - 141°$
tersR.png	ters.png	$111° - 114°$

Table 2. Rotated images, their matches among the template images and the detected rotation angle ranges.

All 15 images are detected correctly with nearly accurate rotation estimations. Also results of repeating the algorithm using more bins and a smaller number of maximum number of peaks are investigated below. First the number of bins has been increased to 180 while keeping other variables fixed. The results of this can be seen from Table 3.

Rotated Book Name	**Matched Template Name**	**Rotation Angle in Degrees**
algorithmsR.png	algorithms.png	$52° - 53°$
bitmemisoykulerR.png	bitmemisoykuler.png	$78° - 79°$
chessR.png	chess.png	$16° - 17°$
cinaliR.png	cinali.png	$20° - 21°$
cppR.png	cpp.png	$19° - 20°$
dataminingR.png	datamining.png	$73° - 74°$
harrypotterR.png	bitmemisoykuler.png	$51° - 52°$
heidiR.png	heidi.png	$168° - 169°$
kpssR.png	kpss.png	$118° - 119°$
lordofringsR.png	lordofrings.png	$174° - 175°$
patternrecognitionR.png	patternrecognition.png	$20° - 21°$
sefillerR.png	patternrecognition.png	$78° - 79°$
shawshankR.png	patternrecognition.png	$125° - 126°$
stephenkingR.png	stephenking.png	$136° - 137°$
tersR.png	ters.png	$109° - 110°$

Table 3. Rotated images, their matches among the template images and the detected rotation angle ranges.

From the results while losing accuracy to 12 correct guesses, the resolution of the estimation angle has got better for some of the images which can be considered as a tradeoff. Next, the number of maximum peaks has been decreased to 200 where less lines are detected and categorized according to their slopes for each image. Result of this operation can be seen in Table 4.

Rotated Book Name	Matched Template Name	Rotation Angle in Degrees
algorithmsR.png	algorithms.png	$54° - 57°$
bitmemisoykulerR.png	bitmemisoykuler.png	$78° - 81°$
chessR.png	chess.png	$18° - 21°$
cinaliR.png	cinali.png	$21° - 24°$
cppR.png	cpp.png	$21° - 24°$
dataminingR.png	datamining.png	$75° - 78°$
harrypotterR.png	stephenking.png	$141° - 144°$
heidiR.png	heidi.png	$168° - 171°$
kpssR.png	kpss.png	$120° - 123°$
lordofringsR.png	lordofrings.png	$174° - 177°$
patternrecognitionR.png	patternrecognition.png	$21° - 24°$
sefillerR.png	sefiller.png	$78° - 81°$
shawshankR.png	shawshank.png	$126° - 129°$
stephenkingR.png	stephenking.png	$138° - 141°$
tersR.png	ters.png	$111° - 114°$

Table 4. Rotated images, their matches among the template images and the detected rotation angle ranges.

From the results the angle values did not change while one of the books are mismatched. The rotation amount of the image "stephenking.png" is similar with the image "harrypotter.png" which might be the reason of the problem. Therefore, decreasing the number of maximum peaks has also caused the performance to be dropped.

Overall the parameters that can give maximum outputs with maximum accuracy are found for this homework writing a good structured code and implementation.

References

[1] MATLAB, "edge," MathWorks, [Online]. Available: https://www.mathworks.com/help/images/ref/edge.html. [Accessed 16 4 2024].

[2] MATLAB, "hough," MathWorks, [Online]. Available: https://www.mathworks.com/help/images/ref/hough.html. [Accessed 16 4 2024].

[3] MATLAB, "houghpeaks," MathWorks, [Online]. Available: https://www.mathworks.com/help/images/ref/houghpeaks.html. [Accessed 16 4 2024].

[4] MATLAB, "houghlines," MathWorks, [Online]. Available: https://www.mathworks.com/help/images/ref/houghlines.html. [Accessed 16 4 2024].