CS 484

Introduction to Computer Vision
Spring 2023

Homework Assignment 2

Yiğit Ekin

21901784

Section 01

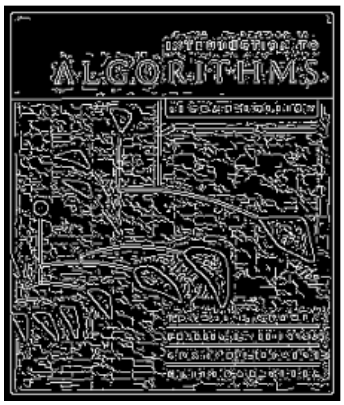
Analysis on Edge Detection:

In order to apply Canny Edge Detection algorithm, a third-party library named openCV is used [1]. The prototype for the used function is the following:

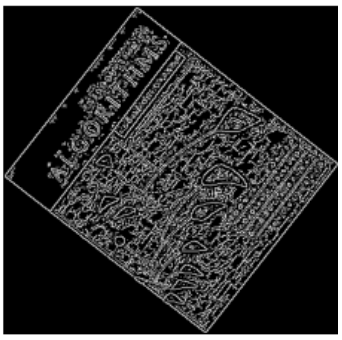
```
Canny(  image,
        threshold1,
        threshold2,
)
```

In the following method, image is the input grayscale image. threshold1 (t1) is the lower bound for the hysteresis and threshold2 (t2) is the upper bound for the hysteresis procedure. While conducting the experiments, the aperture size set as a fixed value of 3. During the experimentations, my first initial thought was to create a low range between t1 and t2 in order to observe what will be the output. Also, in order to observe the effect of t2, a low threshold is selected. This is because, I wanted to observe high number of strong edges with small suppression. So, at first, a (t1, t2) tuple of (10, 20) is used. After that, the results are the following:

Template Image

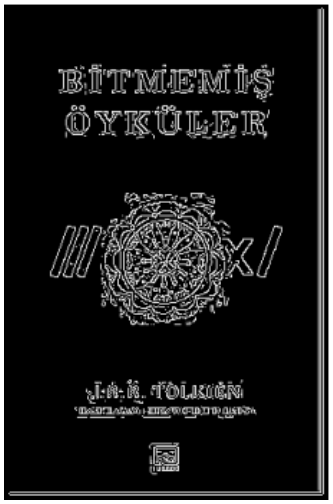


Rotated Image



As it can be seen, although the image does not a have a complex texture, its results is noisy. This is due to low t2 which accepts lots of edges as high.

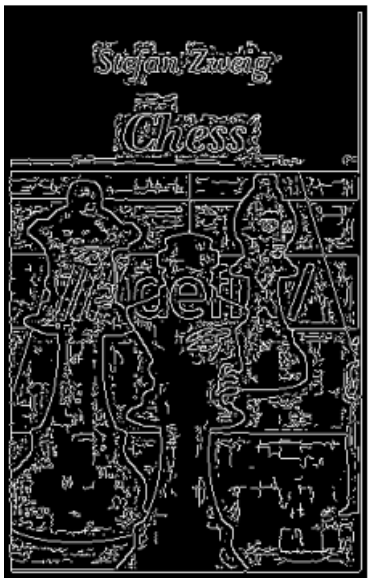
Template Image



Rotated Image

As it can be seen, as the image has low details, its result is sufficiently good because there are a smaller number of false edges to detect.

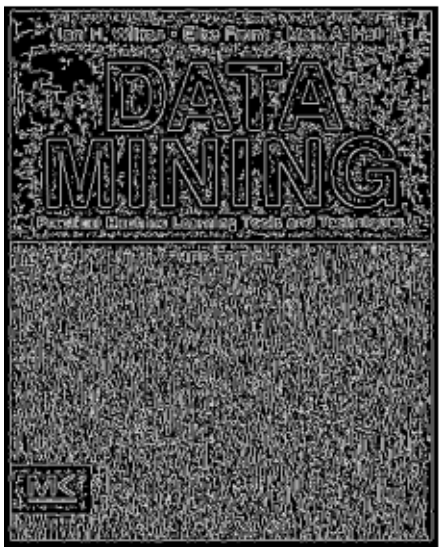
Template Image



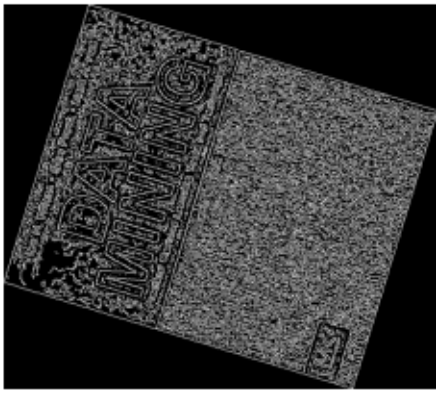
Rotated Image

As it can be seen, although the image does not a have a complex texture, its results is noisy. This is due to low t2 which accepts lots of edges as high.

Template Image

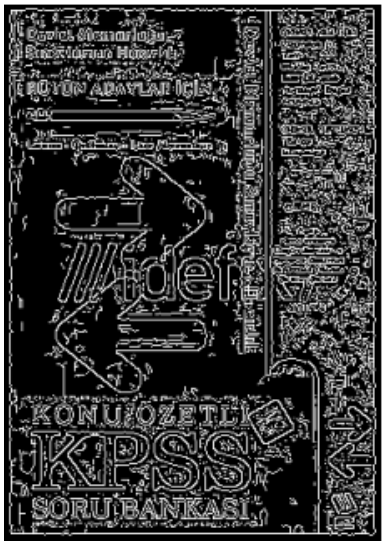


Rotated Image

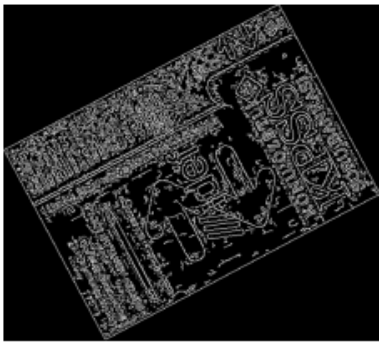


As it can be seen, the image have a complex texture and low t2 accepts lots of edges, its results is noisy.

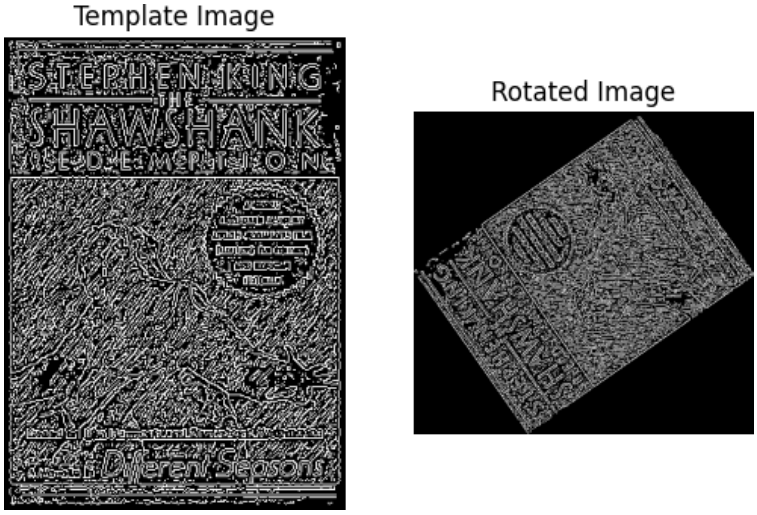
Template Image



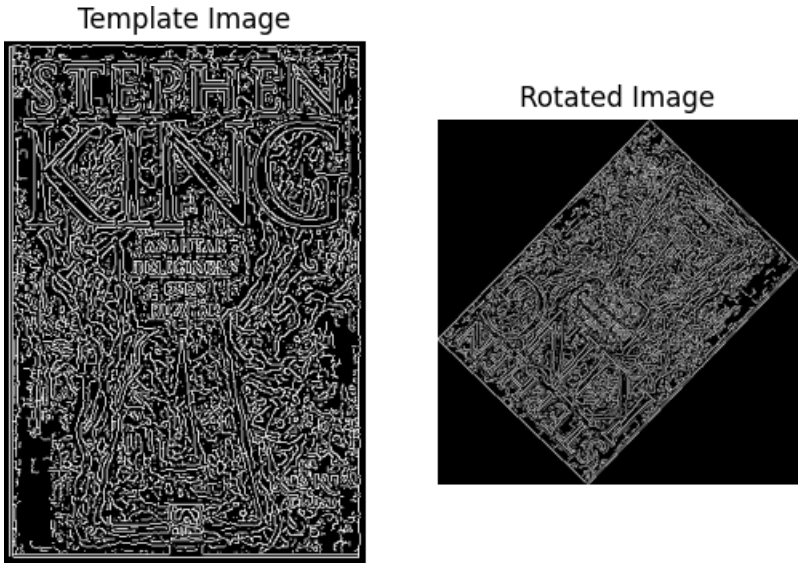
Rotated Image



As it can be seen, although the image does not a have a complex texture, its results is noisy. This is due to low t2 which accepts lots of edges as high.



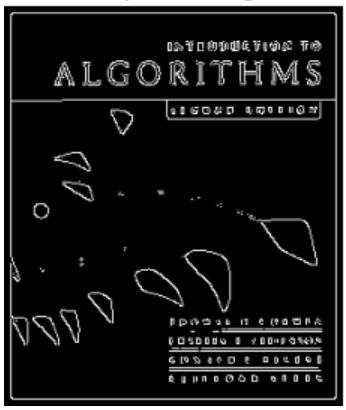
As it can be seen,the image have a complex texture and low t2 accepts lots of edges, its results is noisy.



As it can be seen, the image has a complex texture and low t2 accepts lots of edges, its results is noisy.

Overall, as t2 is selected as low, there were too many edges and the resulted image is too noisy. Although low detailed images are again sufficiently good such as Bitmemiş Öyküler, high detailed images such as Stephen King, Shawshank Redemption and Data Mining are too noisy. To decrease this, (with again low range between t1 and t2) a (t1, t2) tuple of (215, 225) is selected. The results are the following:

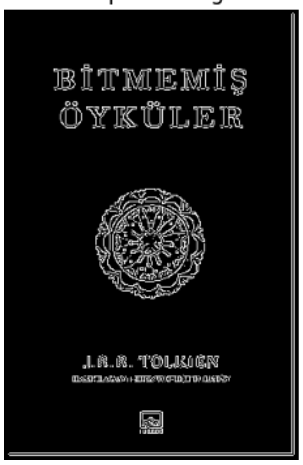
Template Image



Rotated Image

Higher t2 accepted a smaller number of strong edges. Hence, the noise is gone. However, it can be seen that some of the details have been lost due to low t2.

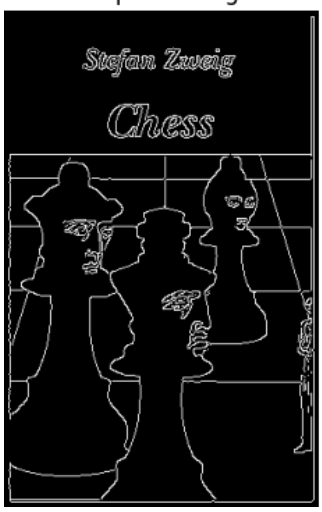
Template Image



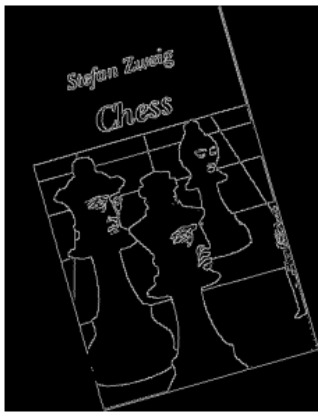
Rotated Image

As there were small noise, there is still small noise as there are less number of false strong edges.

Template Image

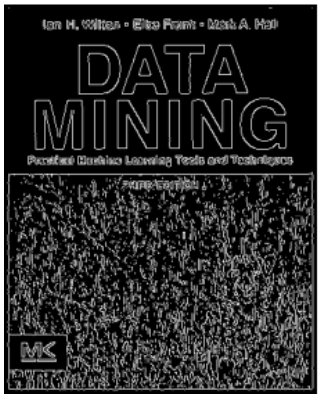


Rotated Image



Higher t2 accepted a smaller number of strong edges. Hence, the noise is gone. However, it can be seen that some of the details have been lost due to low t2.

Template Image



Rotated Image

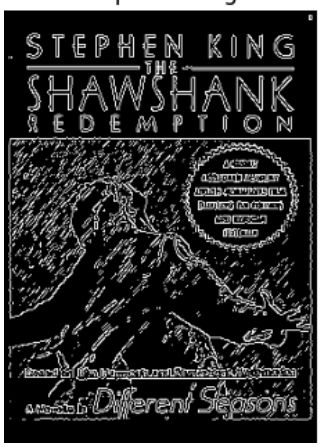
Due to high t2, false strong edges are now smaller. Hence, there are a smaller number of noise presented in the image and the edges are more clear.

Template Image

Rotated Image

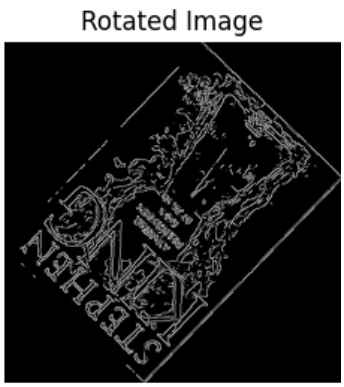
Higher t2 accepted a smaller number of strong edges. Hence, the noise is gone. However, it can be seen that some of the details have been lost due to low t2.

Template Image



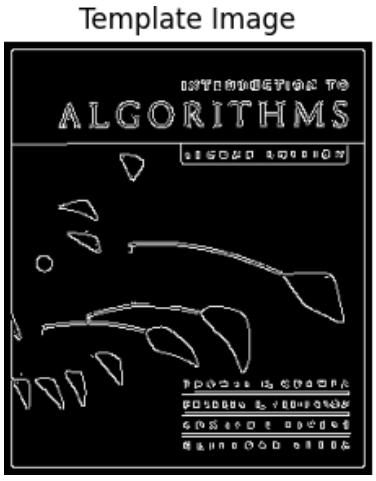
Rotated Image

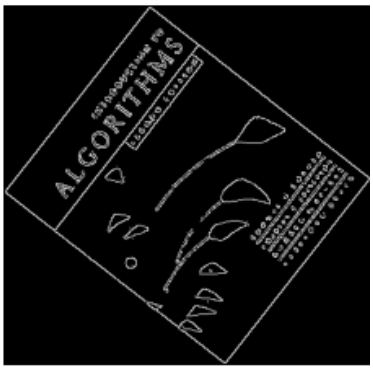
Due to high t2, false strong edges are now smaller. Hence, there are a smaller number of noise presented in the image and the edges are clearer.



Due to high t2, false strong edges are now smaller. Hence, there are a smaller number of noises presented in the image and the edges are clearer.

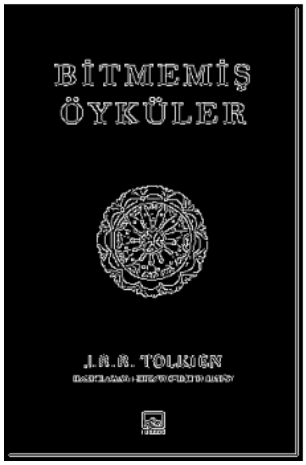
Overall, higher threshold resulted in a better number of edges. This is because a smaller number of edges are classified as a strong edge for the canny edge detector. Hence, noisy images in small t2 are now less noisy and sufficiently good. Hence, it is concluded that high threshold is better for decreasing noise on high feature images. Then, to see the effect of high range between t1 and t2, a tuple of (t1,t2) = (120, 225) is selected. The results are the following:





Due to low t1, some of the lost details are now included.

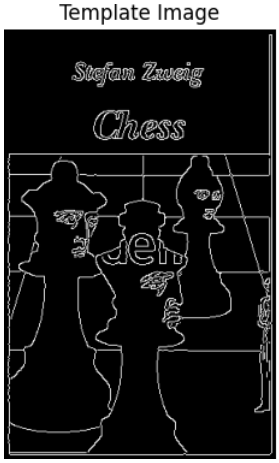
Template Image



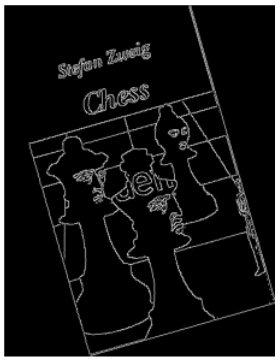
Rotated Image

No crucial effect because there are small number of lost details in the previous one.

Template Image

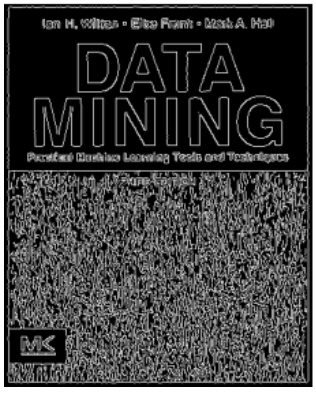


Rotated Image



Due to low t1, some of the lost details are now included.

Template Image



Rotated Image

Due to low t1, some of the lost details are now preserved.

Template Image

Rotated Image

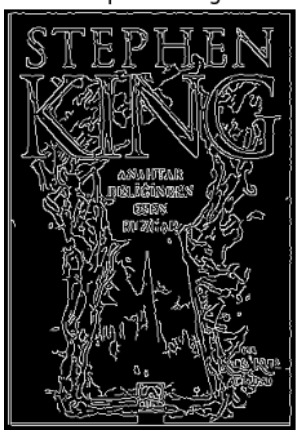
Due to low t1, some of the lost details are now preserved such as the border on the right.

Template Image

Rotated Image

Due to low t1, some of the lost details are now preserved in the middle part.

Template Image



Rotated Image

Due to low t1, some of the lost details are now preserved in the middle part.

Overall, from using low range between t1 and t2, some of the details were lost. Now with allowing some range for hysteresis procedure, these details are now detected. For example, the border that divides the page in the 2 in kpss book is now visible, in pattern recognition book the circles on the right side of the page is now complete, on chess book, the writing that is presented in the middle pawn is now visible and finally in the algorithms book, the tails of the shapes in the middle of the page are now connected. After conducting the following experiments, the consensus for t1 and t2 are selected as 120 and 225. The gatherings of the experiments can be summarized as follows:

- Low t2 can lead to noisy edges in high detail images
- High t2 with low t1 can lead to lose of some relevant edges
- High t2 with sufficiently large t1 is the ideal procedure
- One should aim for high t2 low t1 with the idea of noise in mind

Analysis on Line Detection:

In order to apply Hough Transform algorithm, a third-party library named openCV is used [2]. The prototype for the used function is the following:

```
HoughLinesP(  image,
              lines,
              rho,
              theta,
              threshold,
              minLineLength = 0,
              maxLineGap = 0
            )
```

- edges: Output of the edge detector.
- Image: input image
- lines: A vector to store the coordinates of the start and end of the line.
- rho: The resolution parameter rho in pixels.
- theta: The resolution of the parameter theta in radians.
- threshold: The minimum number of intersecting points to detect a line.
- minLineLength: #minLineLength: Minimum length of line. Line segments shorter than this are rejected.
- maxLineGap: Maximum allowed gap between line segments to treat them as single line.

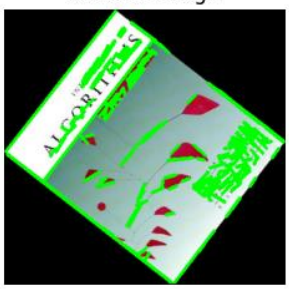
First aim was to observe the importance of threshold. My hypothesis was that low threshold will lead to more line detection and will output noisy lines. To observe this, first the following parameters are used:

rho=1, theta=np.pi/180, threshold=10, min_line_len=10, max_line_gap=10

results are the following:

Template Image

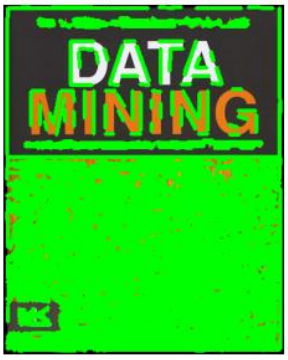
Rotated Image



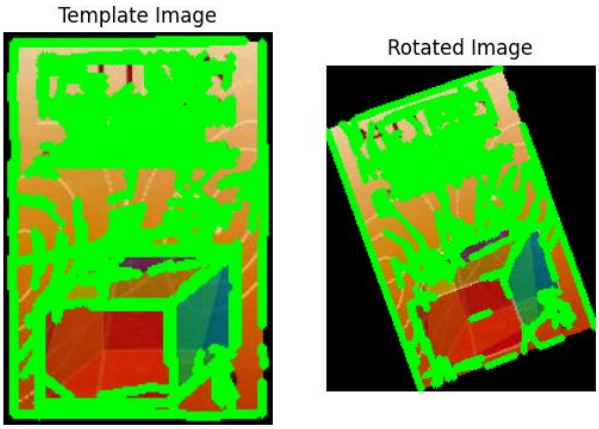
Template Image

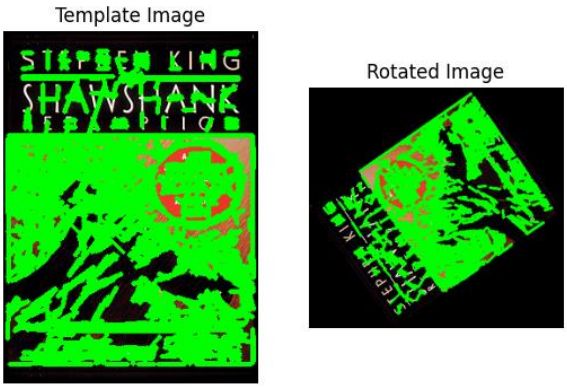
Rotated Image

Template Image



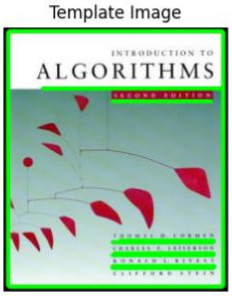
Rotated Image



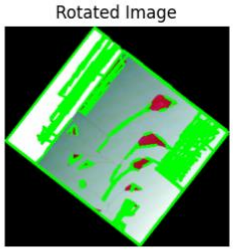


As it can be seen, low threshold has yielded in too many lines. In hough transform, our aim is to find lines that fit the most. This selection was too low and as a result produced too many hough lines especially on images with high details such as pattern recognition shawshank redemption. To solve this, with other parameters kept equal, the threshold is increased up to 60. The results are the following:

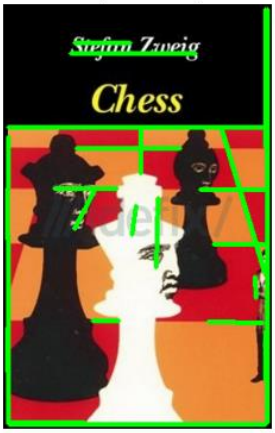
Template Image



Rotated Image

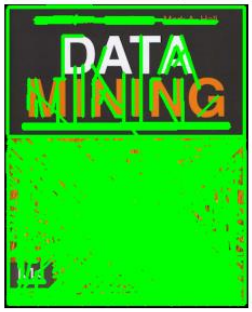


Template Image

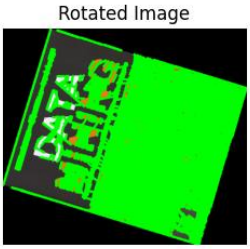


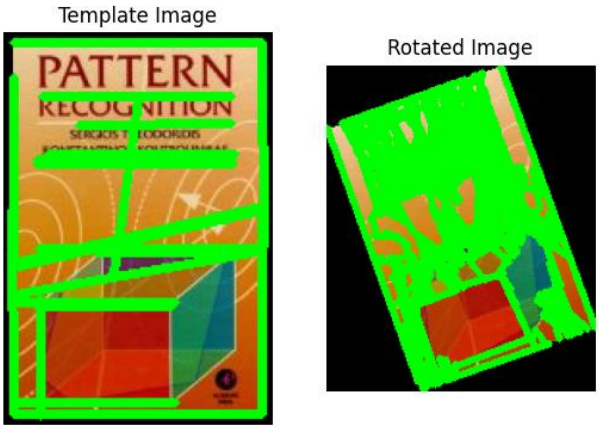
Rotated Image

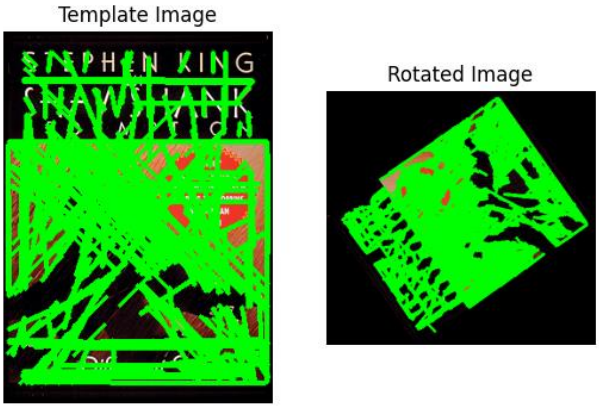
Template Image



Rotated Image

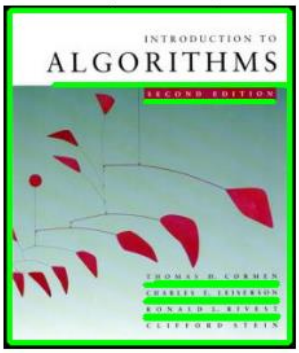




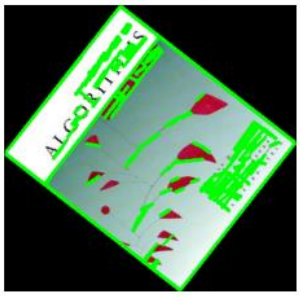


As it can be seen, this resulted in lower number of lines. These are better representation of the images. Although high detail images such as data mining still has lots of lines, unnecessary lines on pattern recognition (without rotation) have been lost. One problem is that rotation makes the lines closer to each other and gives an advantage for line fitting over normal images due to the current selection of max line gap. To solve this problem max line gap is decreased to 5. The results are the following:

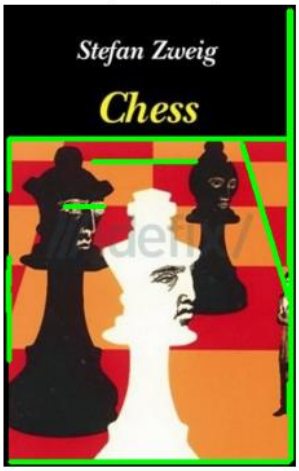
Template Image



Rotated Image

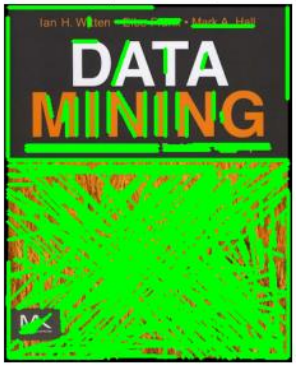


Template Image

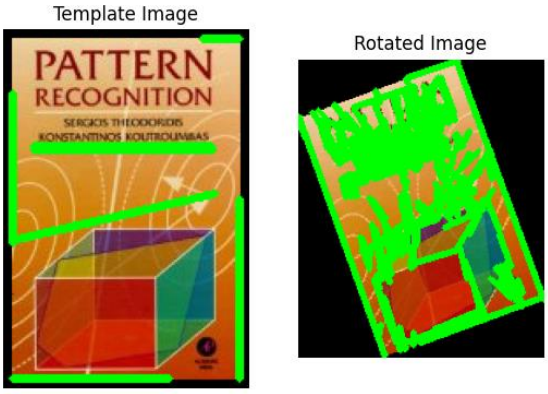


Rotated Image

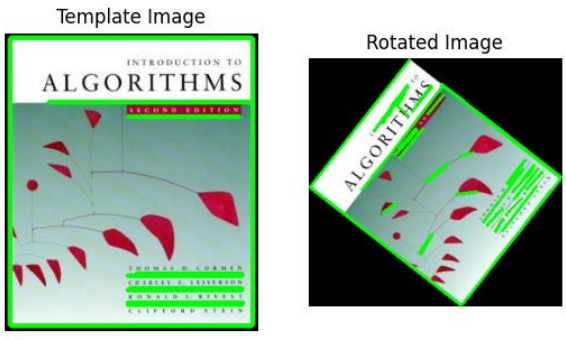
Template Image



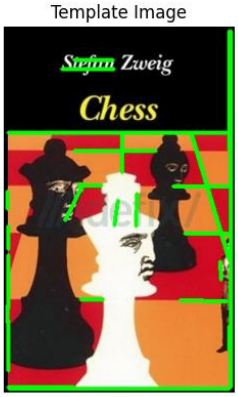
Rotated Image



As it can be seen, this further decreased the number of lines which can be said as a better representation. Also, the noise difference between rotated and normal images have been decreased. Finally, in order to get more confident lines, the min line length have been increased up to 20. The aim is the get more longer and as a result more voted and confident lines which can be interpreted as a better representative candidate.

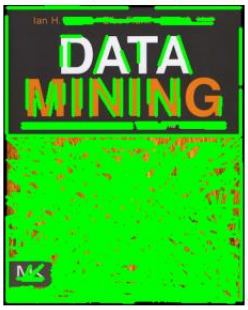


Template Image

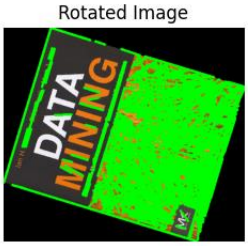


Rotated Image

Template Image



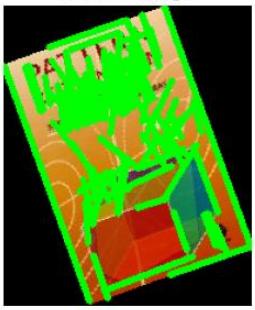
Rotated Image

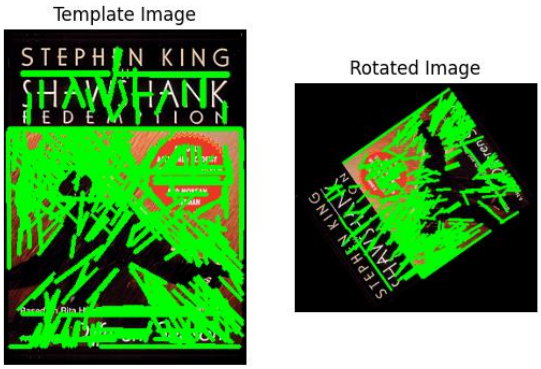


-

Template Image

Rotated Image





Although the changes are not as satisfactory as expected, this parameter selection has yielded a better percentage of histogram matching in later sections. Hence, in the following bins calculation section, the parameters of calculation are: threshold=60, min_line_len=20, max_line_gap=5

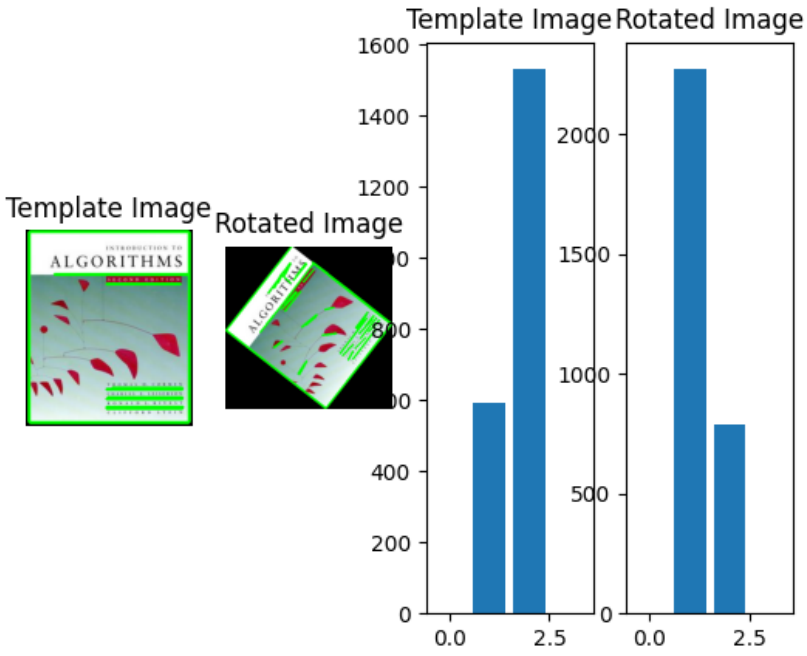
Number of Bin Selection:

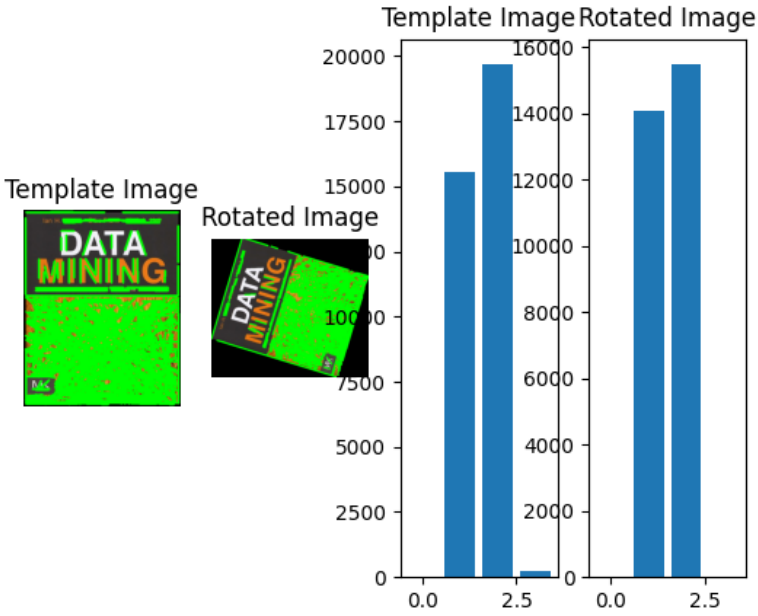
The number of bins correspond to the matching degree of one bin. The formula is the following:

$$degree\_of\_one\_bin = \frac{360}{number\_of\_bins}$$

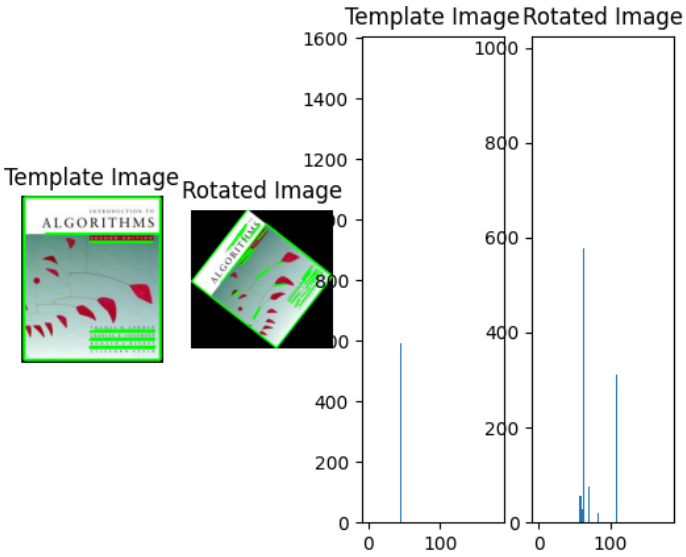
From the formula, it can be seen that high number of bins correspond to a better precision of angle while lower number of bins correspond to a lower precision of bin. The tradeoff is that while calculating the Euclidian distance, too high of precision can lead to overfitting and as a result lower number of matches in the following step. On the other hand, low precision of bin can lead to underfitting. In other words, according to my hypothesis, 45 degrees for 1 bin can be too generic and 2 degrees for a bin can be too specific. To conclude this, the following number of bins were tried and their results are the following:

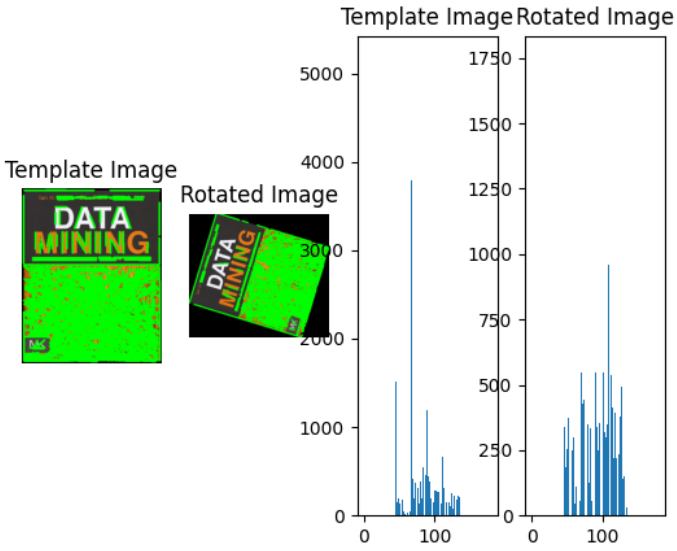
Number of bins = 4, angle of one bin = 90:



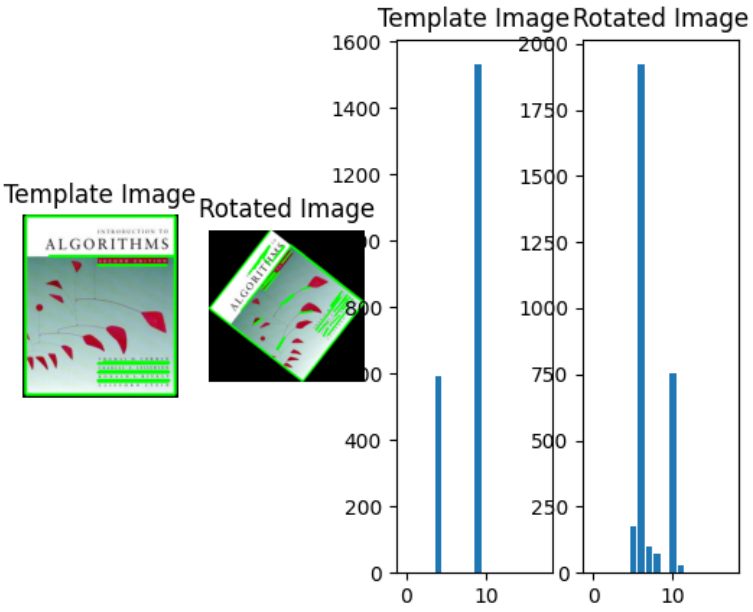


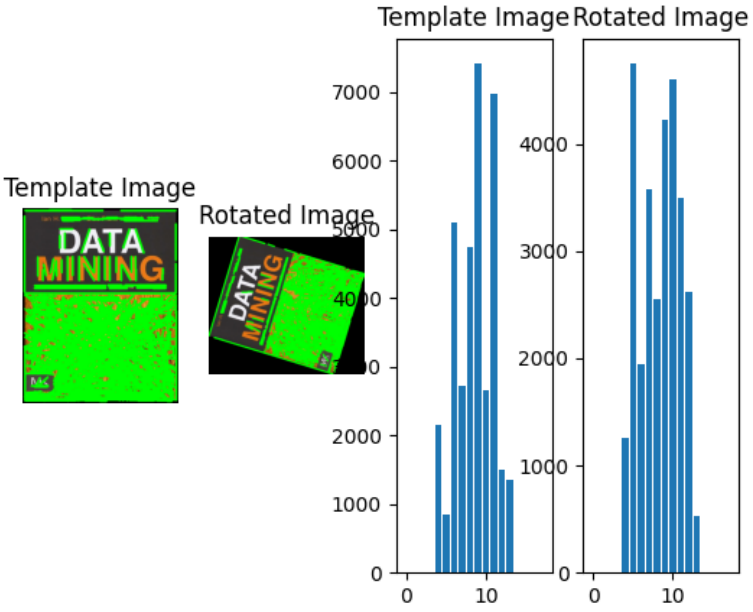
Number of bins = 180, angle of one bin = 2:





Number of bins = 18, angle of one bin = 20:





Matching line orientation histograms:

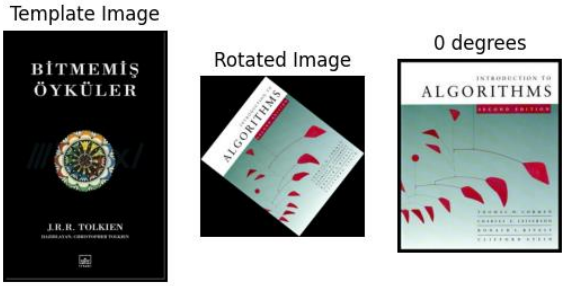
After several experiments, the best accuracy (47% or 7/15) in terms of matching line orientation histograms is done with the following parameters:

Canny: t1 = 120, t2 = 225

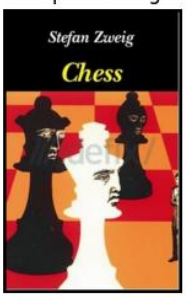
Hough: threshold=60, min_line_len=20, max_line_gap=5

Number_of_bins = 18

In each row of the following results, first image is the original image, the second image is the rotated version of the original image, the third image is the matched imaged with rotated as the corresponding angle calculated from previous steps. Further information can be included from the jupyter notebook file attached in the submission. According to my analysis, some of the images are rotated 180 degrees more than needed. This can be resulted due to our histogram. Line orientation histograms have no information regarding direction while shifting. In other words, while calculating the Eucledian distance, shifting 30 degrees and 150 degrees have no difference.



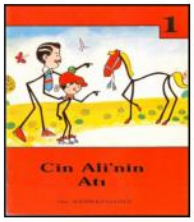
Template Image



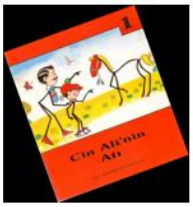
Rotated Image

20 degrees

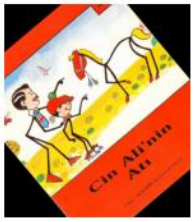
Template Image



Rotated Image

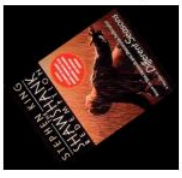


40 degrees

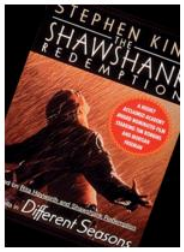


Template Image

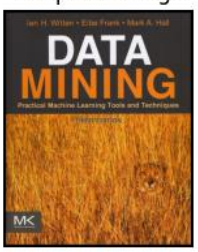
Rotated Image



340 degrees

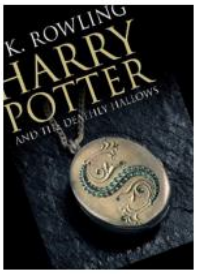


Template Image

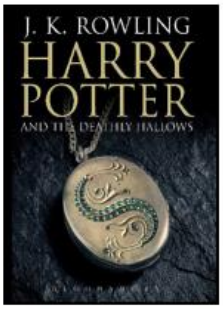


Rotated Image

20 degrees



Template Image



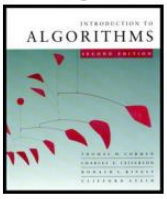
Rotated Image

120 degrees

Template Image

Rotated Image

0 degrees

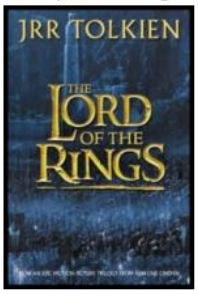


Template Image

Rotated Image

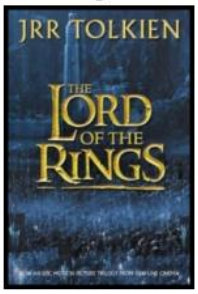
300 degrees

Template Image

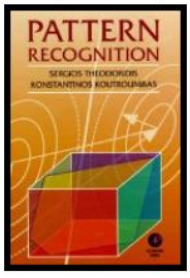


Rotated Image

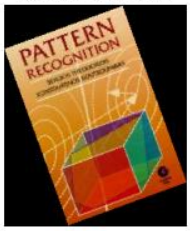
0 degrees



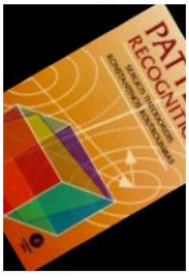
Template Image



Rotated Image

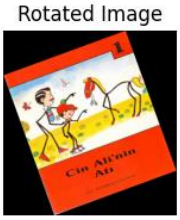


300 degrees

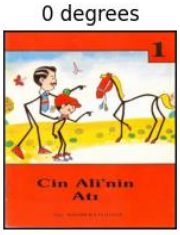


Template Image

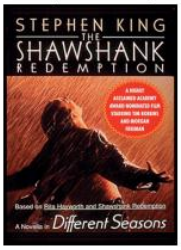
Rotated Image



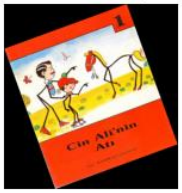
0 degrees



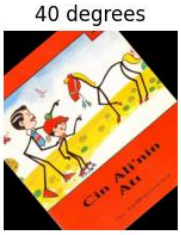
Template Image



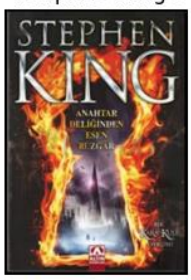
Rotated Image



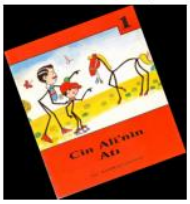
40 degrees



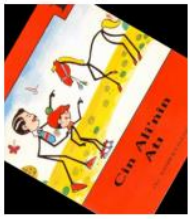
Template Image



Rotated Image



60 degrees



Template Image

Rotated Image

300 degrees

References:

[1] “Feature detection,” *OpenCV*. [Online]. Available: https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga04723e007ed888ddf11d9ba04e2232de. [Accessed: 26-Apr-2023].

[2] “Hough Line transform,” *OpenCV*. [Online]. Available: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html. [Accessed: 26-Apr-2023].