CS484
Homework Assignment 1

Boran Kılıç
22103444

1. Line Based Object Matching

The aim of this assignment was to identify objects by analyzing and comparing their line patterns. Specifically, we attempted to match books based on the line orientation histograms derived from images of their covers.

1.1 Edge detection:

First we applied edge detection to all the images. In order to do this the OpenCV library has been used [1].

The edge detection code involves several parameters:

Gaussian Blur Parameters:

- The size of the Gaussian kernel, which is set to 15x15 pixels, determining the extent of the blur.
- 1.4: The standard deviation in the X and Y directions for the Gaussian kernel, controlling the amount of blur.

Canny Edge Detection Parameters:

If a pixel's gradient exceeds the upper threshold, it is considered an edge. Conversely, if the pixel's gradient is below the lower threshold, it is discarded.

- The lower threshold = 1
- The upper threshold = 50
- apertureSize = 3: The size of the Sobel kernel used for gradient calculation, set to 3x3.
- L2gradient = True: Indicating that the more accurate L2 norm is used to compute the gradient magnitude instead of the L1 norm.

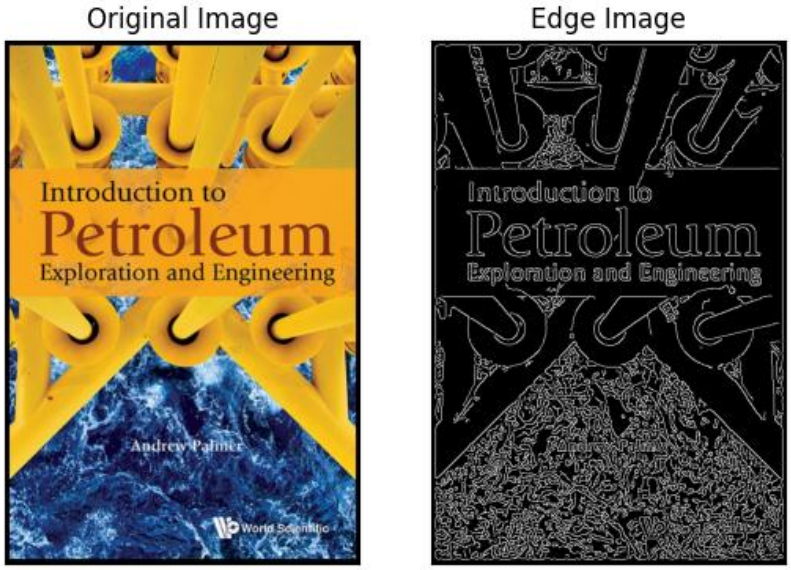


Figure 1.1: Edge detection $th_{low} = 1$, $th_{high} = 50$

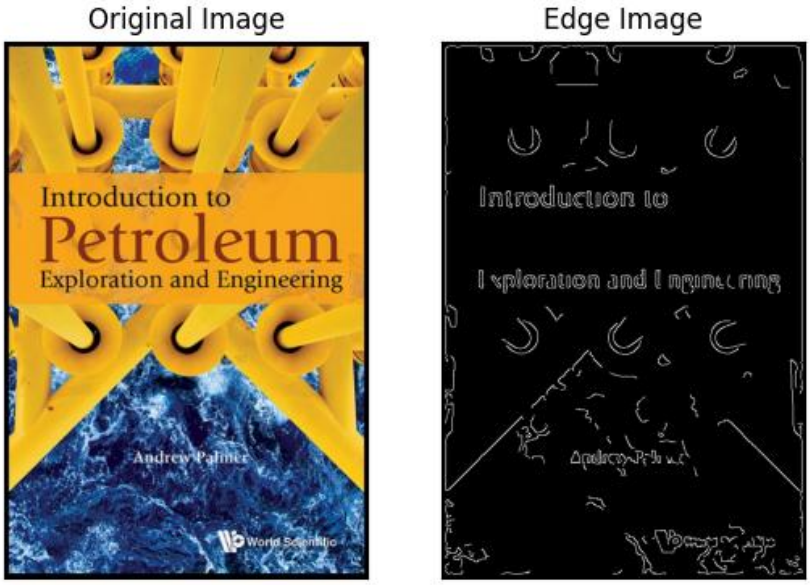


Figure 1.2: Edge detection $th_{low} = 100$, $th_{high} = 150$

The results of the Canny edge detection process depend on the choice of thresholds and the Gaussian blur parameters. When the lower and upper thresholds for edge detection are too close, the algorithm may not identify significant edges, leading to a sparse result. Conversely, setting the thresholds too far apart can result in noisy edges, where irrelevant details are detected. The size of the Gaussian kernel also plays a critical role in smoothing out the image, larger kernel sizes tend to blur the image more, reducing high-frequency noise but also possibly losing fine details in the edges. As the kernel size increases, the edges may become smoother, and fewer noise artifacts will be detected, but at the

cost of potentially missing small or intricate edges. Adjusting these parameters allows for a balance between edge sensitivity and noise suppression.

1.2 Hough transform:

The second task was to apply Hough Transform to the detected edges and displaying the lines on the original images. OpenCV library has been used for Hough Transform [2].

The Hough Transform code involves several parameters:

The distance resolution in pixels. This parameter specifies the resolution of the accumulator array in terms of pixel distance. A value of 2 means the resolution is 2 pixels.

`np.pi/180`: The angle resolution in radians. This specifies the granularity of the angle steps in the accumulator space. Here, it's set to one degree (π/180 radians).

`threshold=50`: The minimum number of votes required for a line to be considered valid. A higher threshold means only the strongest lines will be detected.

`minLineLength=150`: The minimum allowed length of a detected line. Lines shorter than this value will not be considered.

`maxLineGap=15`: The maximum allowed gap between segments of a line for them to be connected. If two line segments are within this gap, they are considered part of the same line.

Some sample Hough Transforms along with their histograms are provided below.

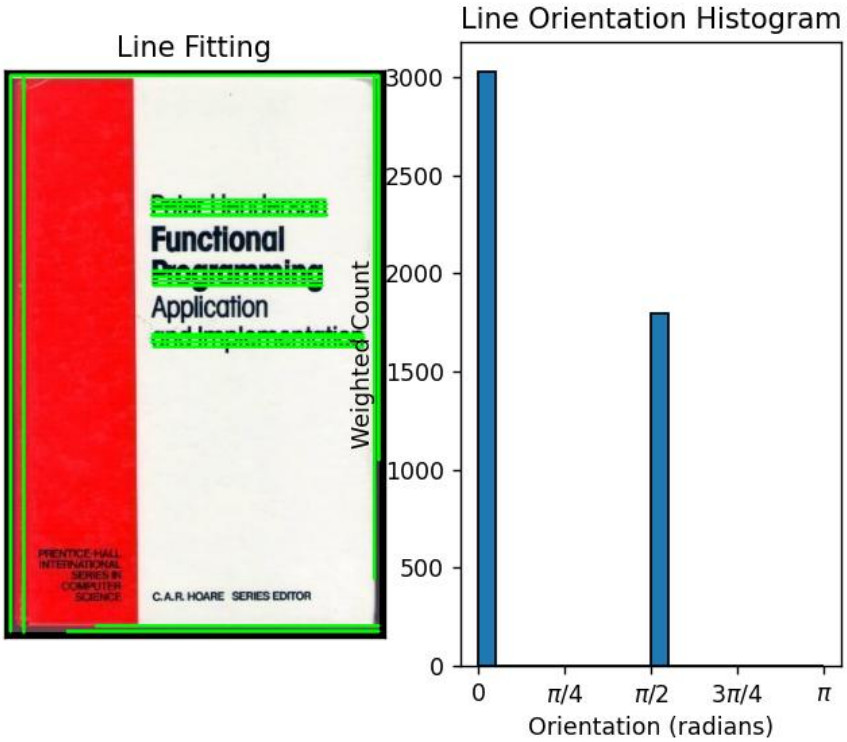


Figure 1.3: Line Fitting and Histogram of "fn.jpeg"

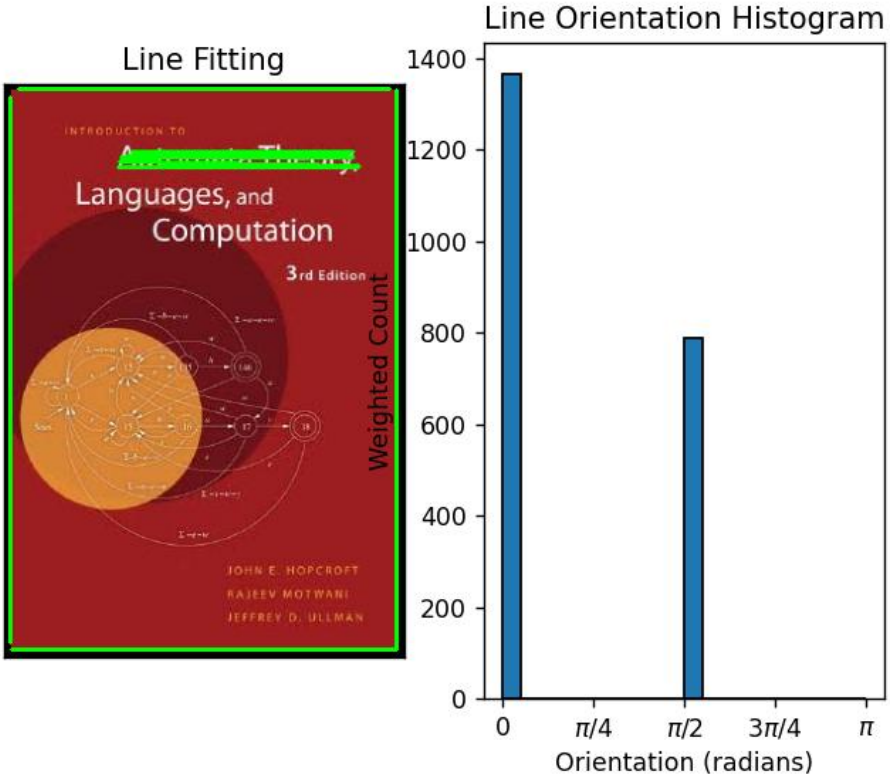


Figure 1.4: Line Fitting and Histogram of "automata.jpeg"

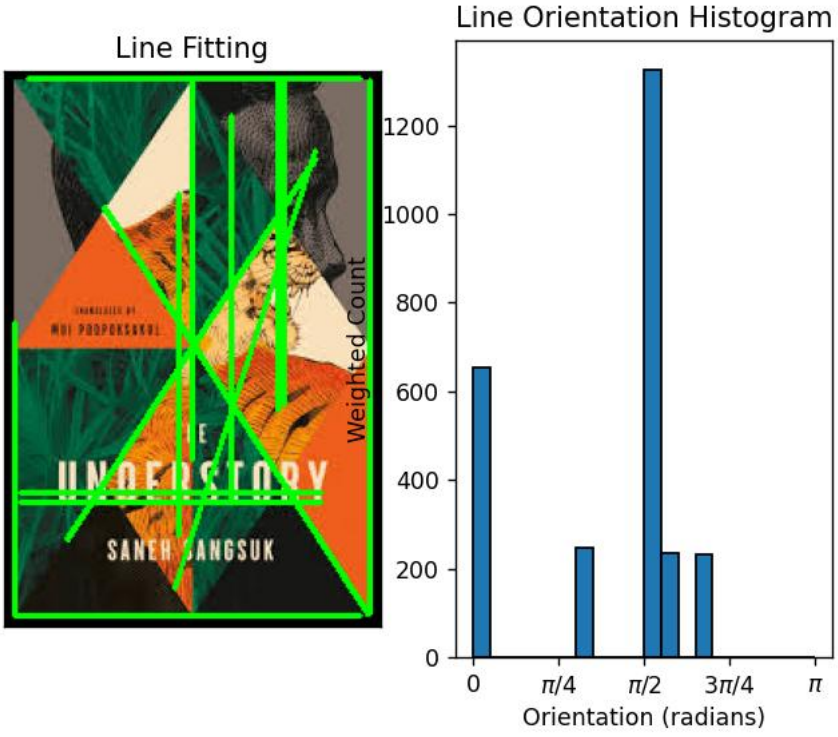


Figure 1.5: Line Fitting and Histogram of "understory.jpeg"

When applying the Hough Transform to detect lines, the minLineLength and maxLineGap parameters are essential in determining the completeness and continuity of detected lines. Increasing the minimum line length reduces the number of smaller, potentially irrelevant lines, which can improve the performance by focusing only on significant structures in the image. However, setting this parameter too high can cause valid but shorter lines to be excluded. The maxLineGap parameter controls how close two line segments must be to be considered part of the same line. A smaller gap results in fewer, more connected lines, whereas a larger gap may result in fragmented lines being grouped together. Experimenting with these values helps refine the detection, affecting the accuracy of line representation.

1.3 Histogram:

The histograms are plotted using different number of bins. The number of bins are 50, 20 and 5.

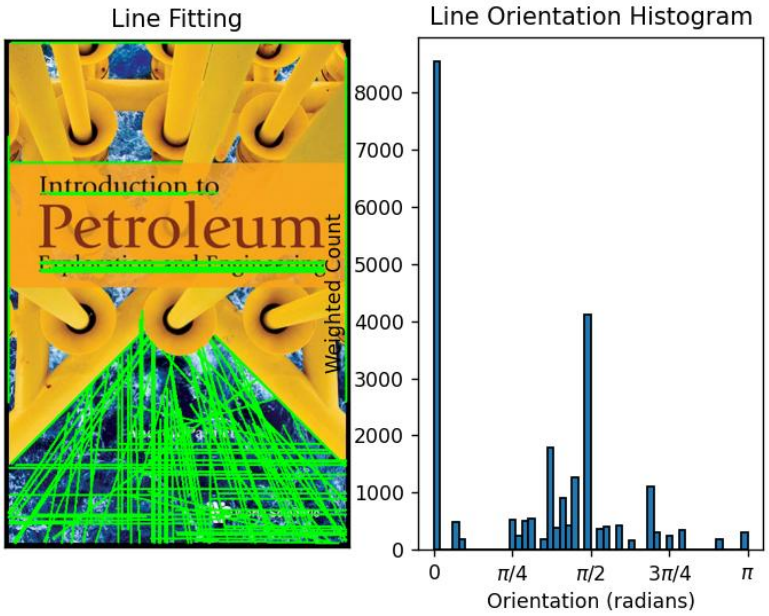


Figure 1.6: Histogram with bin number = 50

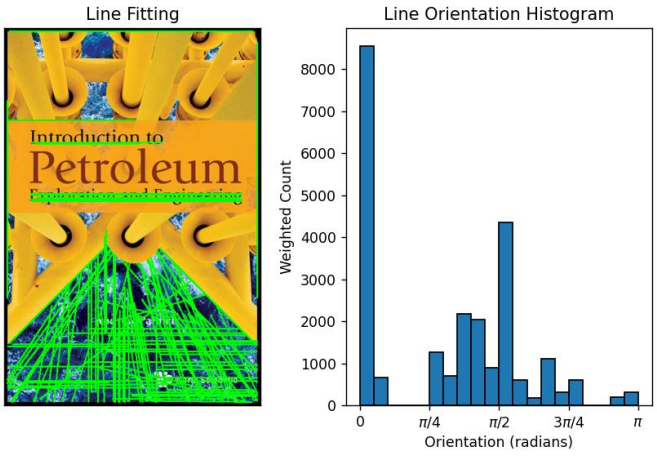


Figure 1.7: Histogram with bin number = 20

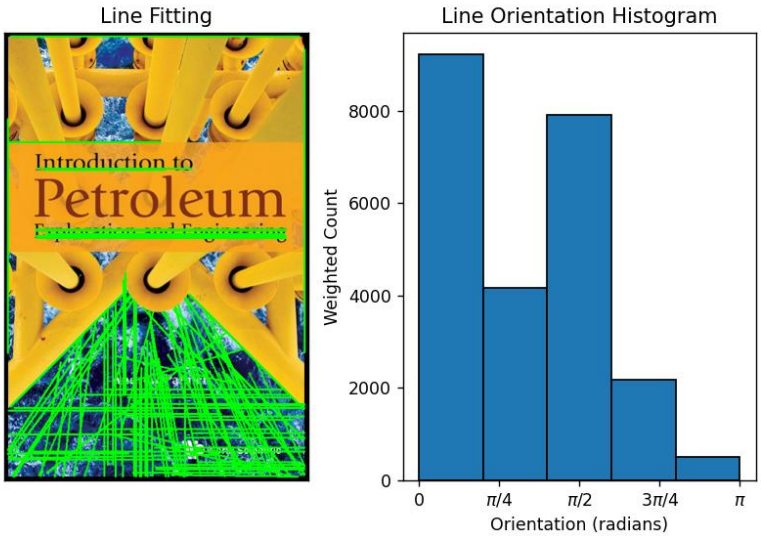


Figure 1.8: Histogram with bin number = 5

Increasing the number of bins also increases the computational cost, as the algorithm has to evaluate more possible line parameters. Conversely, using fewer bins can speed up computation but might reduce the accuracy of line detection, potentially leading to the merging of distinct lines or missing finer details. Therefore, choosing the right number of bins is a trade-off between computational efficiency and the precision of detected lines. In addition, the number of bins also affects the matching process because some lines should be grouped together in order to get more accurate results. When the number of bins were 50 and 5 we have seen less matches than the number of bins were 20. The regarding results are given below in Figures 9-11.

```
Rotated book 'rotated_automata.jpeg' matches with original book 'noname.png' with an angle of rotation of 165.60 degrees.
Rotated book 'rotated_c.jpeg' matches with original book 'c.jpeg' with an angle of rotation of 133.20 degrees.
Rotated book 'rotated_cleancode.jpeg' matches with original book 'cv.jpeg' with an angle of rotation of 72.00 degrees.
Rotated book 'rotated_cs.jpeg' matches with original book 'cs.jpeg' with an angle of rotation of 180.00 degrees.
Rotated book 'rotated_cv.jpeg' matches with original book 'cleancode.jpeg' with an angle of rotation of 43.20 degrees.
Rotated book 'rotated_fn.jpeg' matches with original book 'understory.jpeg' with an angle of rotation of 165.60 degrees.
Rotated book 'rotated_machineage.jpeg' matches with original book 'cv.jpeg' with an angle of rotation of 14.40 degrees.
Rotated book 'rotated_noname.png' matches with original book 'noname.png' with an angle of rotation of 14.40 degrees.
Rotated book 'rotated_os.jpeg' matches with original book 'c.jpeg' with an angle of rotation of 133.20 degrees.
Rotated book 'rotated_petrol.jpeg' matches with original book 'strip.png' with an angle of rotation of 93.60 degrees.
Rotated book 'rotated_proofs.jpeg' matches with original book 'topology.jpeg' with an angle of rotation of 104.40 degrees.
Rotated book 'rotated_soul.jpeg' matches with original book 'machineage.jpeg' with an angle of rotation of 57.60 degrees.
Rotated book 'rotated_strip.png' matches with original book 'cv.jpeg' with an angle of rotation of 57.60 degrees.
Rotated book 'rotated_topology.jpeg' matches with original book 'understory.jpeg' with an angle of rotation of 46.80 degrees.
Rotated book 'rotated_understory.jpeg' matches with original book 'automata.jpeg' with an angle of rotation of 180.00 degrees.
Number of matches:  3 when num_bins = 50
```

Figure 1.9: Matching and the rotation angle results bin number = 5

```
Rotated book 'rotated_automata.jpeg' matches with original book 'automata.jpeg' with an angle of rotation of 72.00 degrees.
Rotated book 'rotated_c.jpeg' matches with original book 'c.jpeg' with an angle of rotation of 108.00 degrees.
Rotated book 'rotated_cleancode.jpeg' matches with original book 'cleancode.jpeg' with an angle of rotation of 72.00 degrees.
Rotated book 'rotated_cs.jpeg' matches with original book 'cs.jpeg' with an angle of rotation of 180.00 degrees.
Rotated book 'rotated_cv.jpeg' matches with original book 'machineage.jpeg' with an angle of rotation of 36.00 degrees.
Rotated book 'rotated_fn.jpeg' matches with original book 'os.jpeg' with an angle of rotation of 72.00 degrees.
Rotated book 'rotated_machineage.jpeg' matches with original book 'petrol.jpeg' with an angle of rotation of 180.00 degrees.
Rotated book 'rotated_noname.png' matches with original book 'noname.png' with an angle of rotation of 144.00 degrees.
Rotated book 'rotated_os.jpeg' matches with original book 'os.jpeg' with an angle of rotation of 36.00 degrees.
Rotated book 'rotated_petrol.jpeg' matches with original book 'petrol.jpeg' with an angle of rotation of 180.00 degrees.
Rotated book 'rotated_proofs.jpeg' matches with original book 'understory.jpeg' with an angle of rotation of 180.00 degrees.
Rotated book 'rotated_soul.jpeg' matches with original book 'soul.jpeg' with an angle of rotation of 144.00 degrees.
Rotated book 'rotated_strip.png' matches with original book 'cleancode.jpeg' with an angle of rotation of 36.00 degrees.
Rotated book 'rotated_topology.jpeg' matches with original book 'fn.jpeg' with an angle of rotation of 36.00 degrees.
Rotated book 'rotated_understory.jpeg' matches with original book 'proofs.jpeg' with an angle of rotation of 180.00 degrees.
Number of matches:  8 when num_bins = 5
```

Figure 1.10: Matching and the rotation angle results bin number = 50

The histograms are matched using 20 bins as the results were sufficient with this bin number.

```
Rotated book 'rotated_automata.jpeg' matches with original book 'automata.jpeg' with an angle of rotation of 162.00 degrees.
Rotated book 'rotated_c.jpeg' matches with original book 'c.jpeg' with an angle of rotation of 126.00 degrees.
Rotated book 'rotated_cleancode.jpeg' matches with original book 'cleancode.jpeg' with an angle of rotation of 72.00 degrees.
Rotated book 'rotated_cs.jpeg' matches with original book 'cs.jpeg' with an angle of rotation of 180.00 degrees.
Rotated book 'rotated_cv.jpeg' matches with original book 'machineage.jpeg' with an angle of rotation of 36.00 degrees.
Rotated book 'rotated_fn.jpeg' matches with original book 'os.jpeg' with an angle of rotation of 162.00 degrees.
Rotated book 'rotated_machineage.jpeg' matches with original book 'petrol.jpeg' with an angle of rotation of 90.00 degrees.
Rotated book 'rotated_noname.png' matches with original book 'noname.png' with an angle of rotation of 144.00 degrees.
Rotated book 'rotated_os.jpeg' matches with original book 'os.jpeg' with an angle of rotation of 126.00 degrees.
Rotated book 'rotated_petrol.jpeg' matches with original book 'cv.jpeg' with an angle of rotation of 180.00 degrees.
Rotated book 'rotated_proofs.jpeg' matches with original book 'proofs.jpeg' with an angle of rotation of 90.00 degrees.
Rotated book 'rotated_soul.jpeg' matches with original book 'soul.jpeg' with an angle of rotation of 54.00 degrees.
Rotated book 'rotated_strip.png' matches with original book 'cv.jpeg' with an angle of rotation of 54.00 degrees.
Rotated book 'rotated_topology.jpeg' matches with original book 'fn.jpeg' with an angle of rotation of 36.00 degrees.
Rotated book 'rotated_understory.jpeg' matches with original book 'understory.jpeg' with an angle of rotation of 90.00 degrees.
Number of matches:  9
```

Figure 1.11: Matching and the rotation angle results bin number = 20

9 matches out of 15 was the best result that I could get no matter how much I tried to optimize the hyper-parameters. Figures of the resulting matches can be found in the Appendix A.

2. Image Segmentation with Superpixels

2.1 Over-segmentation

The initial step is to achieve an over-segmentation for every image. For this purpose SLIC method has been chosen. The following figures include the over-segmentation results on every image.

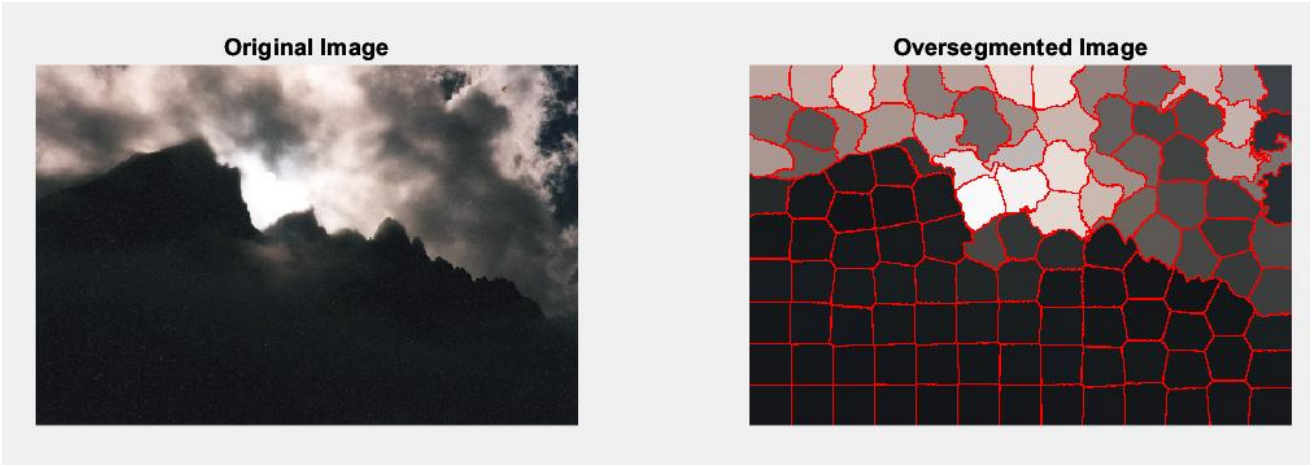


Figure 2.1: Oversegmentation of image 1

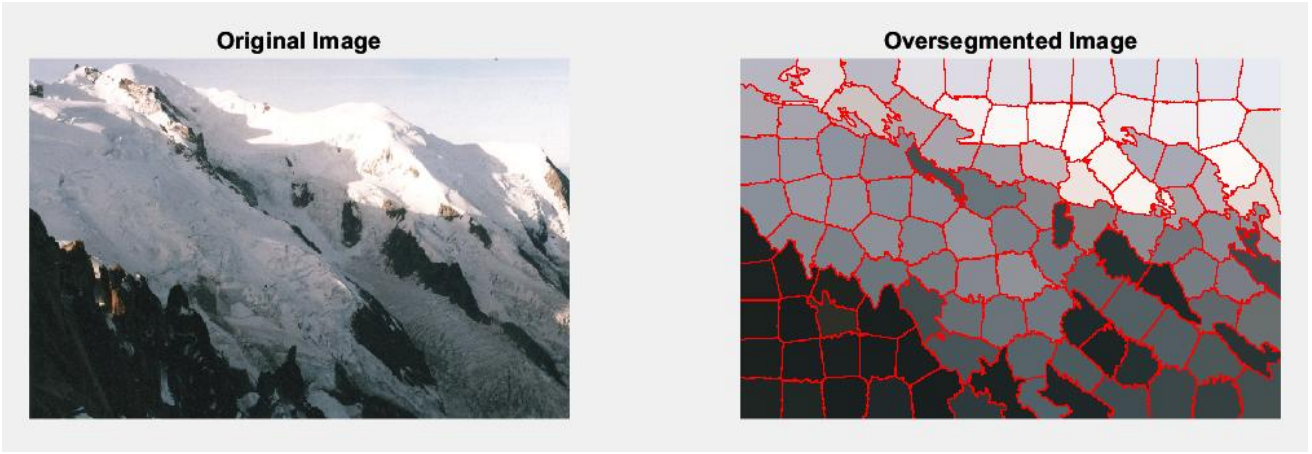


Figure 2.2: Oversegmentation of image 2

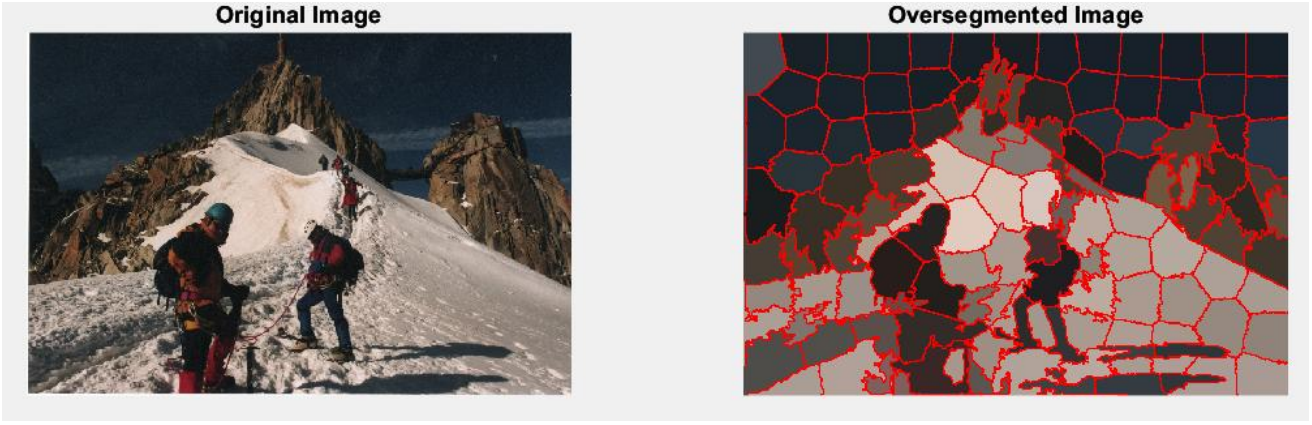


Figure 2.3: Oversegmentation of image 3

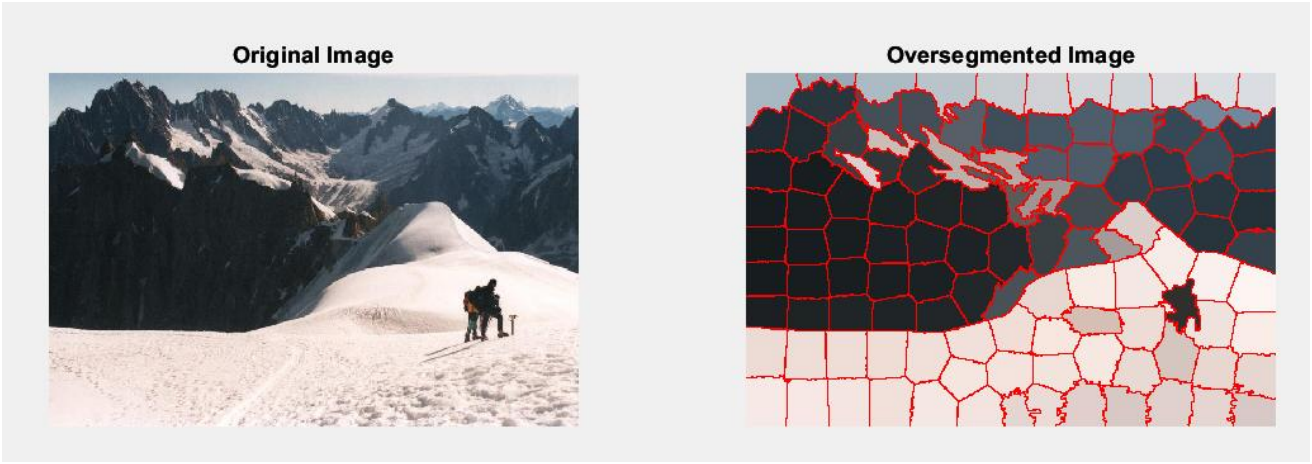


Figure 2.4: Oversegmentation of image 4

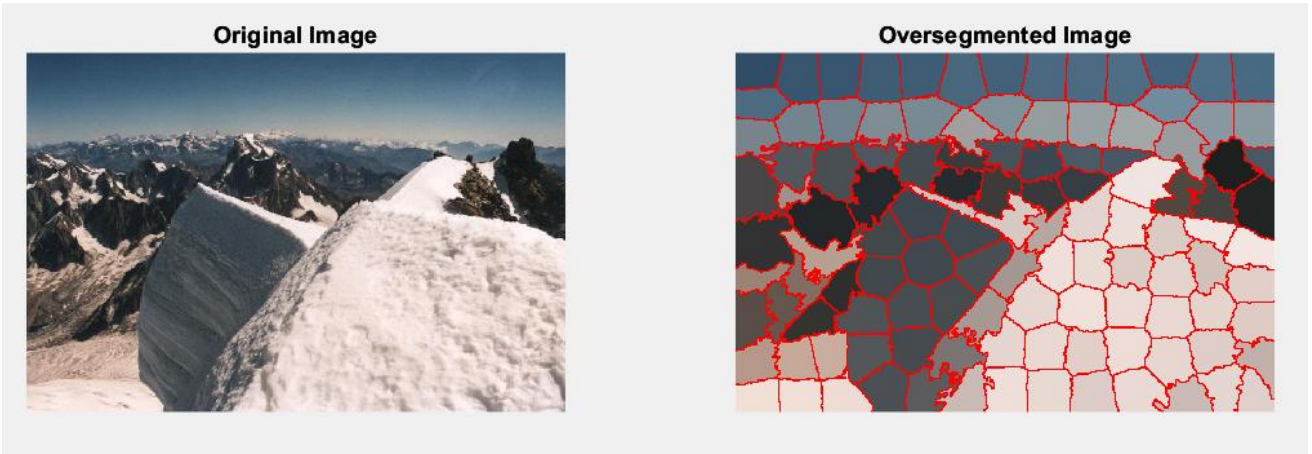


Figure 2.5: Oversegmentation of image 5

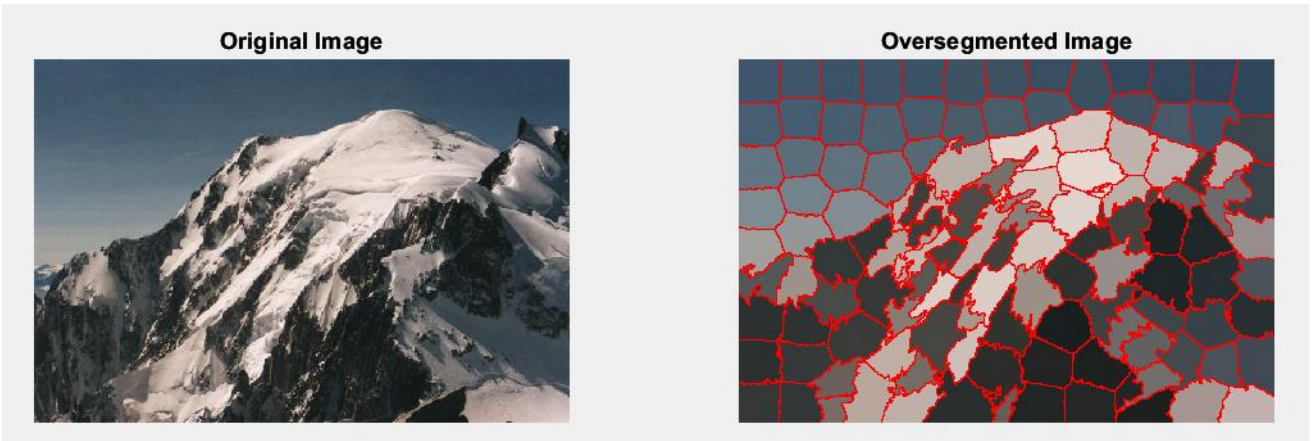


Figure 2.6: Oversegmentation of image 6

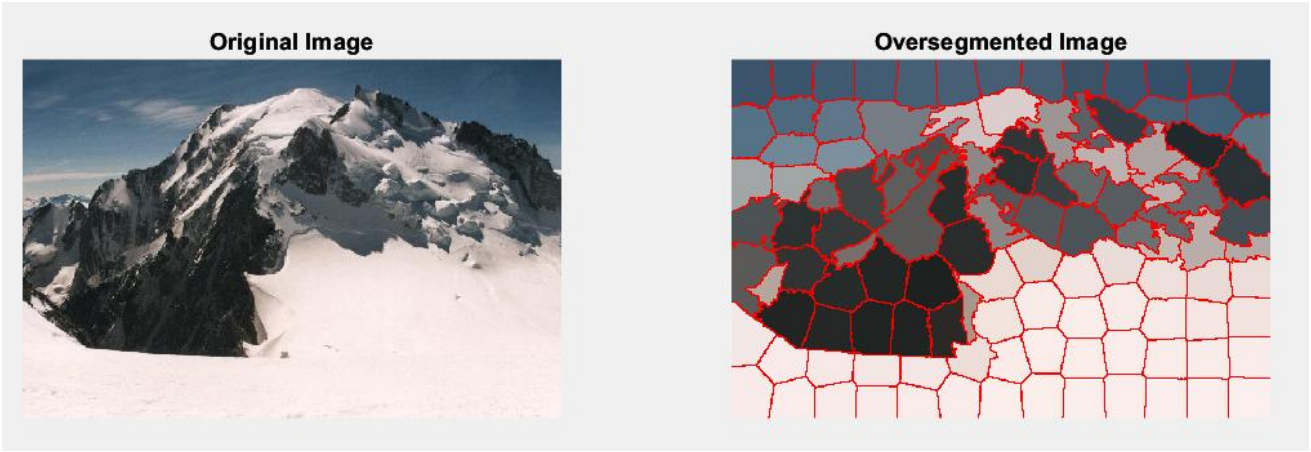


Figure 2.7: Oversegmentation of image 7

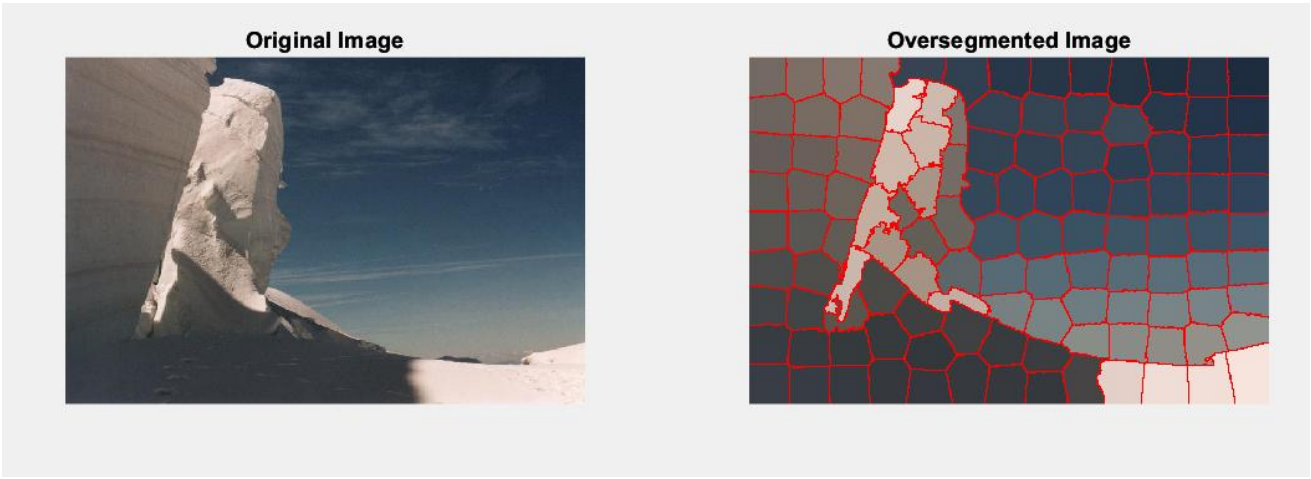


Figure 2.8: Oversegmentation of image 8

Figure 2.9: Oversegmentation of image 9

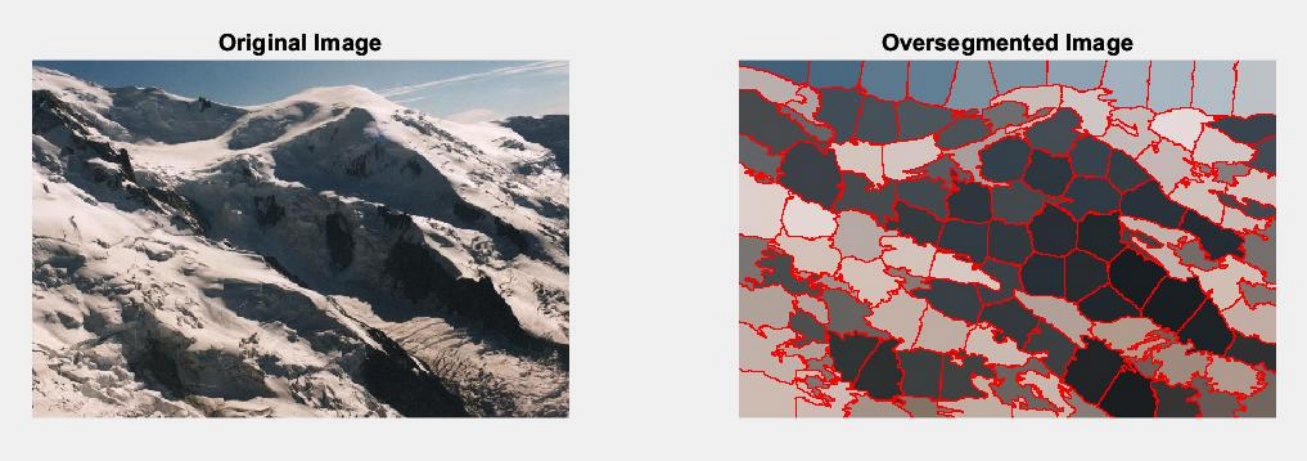


Figure 2.10: Oversegmentation of image 10

The parameters for over segmentation are:

- N as the requested number of super-pixels chosen as 120. This refers to the total number of super-pixels desired in the segmentation process. By setting N=120, we are specifying that the image will be divided into 120 distinct super-pixels. This number influences the granularity of the segmentation; a smaller number will result in larger, coarser segments, while a larger number will produce more fine-grained, detailed super-pixels. The choice of 120 strikes a balance between detail and computational efficiency.
- Compactness: Chosen as 25. This parameter controls the shape and smoothness of the super-pixels. A higher compactness value encourages the creation of more compact, regular-shaped super-pixels, whereas a lower value allows for more irregular and elongated super-pixels. Compactness effectively influences how much the segmentation adheres to the natural boundaries of the objects in the image, rather than over-dividing areas with similar colors or textures.

2.2 Gabor Filter

The next was to computing Gabor texture features for all images. In order to accomplish this task the Matlab implementation of Dr. Peter Kovesi is used [3]. The Gabor filter parameter selections are as follows:

The Gabor filter parameters specified here are designed to extract texture features from an image at multiple scales and orientations. The filter bank is composed of 4 scales and 4 orientations. The **minWaveLength** of 3 determines the wavelength of the smallest scale filter, while the **mult** parameter of 1.7 sets the scaling factor between successive filters, ensuring that each scale captures progressively finer details. The **sigmaOnf** value of 0.65 defines the width of the Gaussian envelope in the frequency domain, which controls the selectivity of the filter. The **dThetaOnSigma** parameter of 1.3 defines the angular spread of the orientations, making the filters sensitive to different spatial frequencies and orientations in the image. By using these parameters, you obtain a well-defined set of Gabor filters that are particularly useful for analyzing texture and frequency information across different image regions. Adjusting these parameters allows for fine-tuning the filter bank to match the characteristics of the texture in the image being analyzed.

```
orientations = [0 pi/4 pi/2 3*pi/4];
scales = [0.25 0.5 0.75 1.0];
```

2.3 Pseudo Colour Image

In that part the superpixels are clustered using k-means algorithm and psudo-color images are generated for each image. **Number of clusters** chosen as 5. The number of clusters determines how finely the image is segmented; a smaller number of clusters results in fewer, larger regions with more generalized texture patterns, while a larger number of clusters creates finer segmentation with more detailed texture distinctions but noisier results.

The resulting images are as follows:

Figure 2.11: Pseudo-Color Image 1

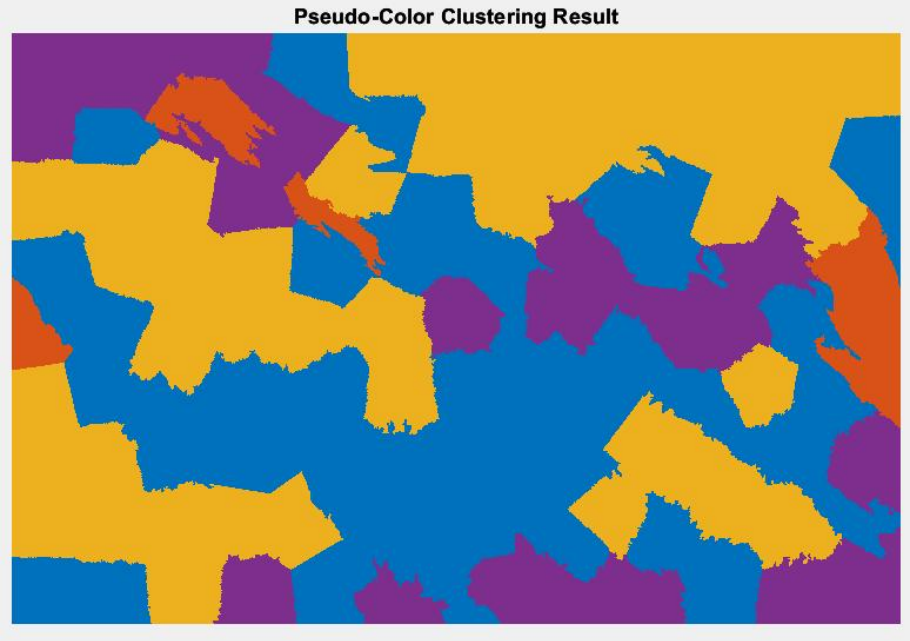


Figure 2.12: Pseudo-Color Image 2

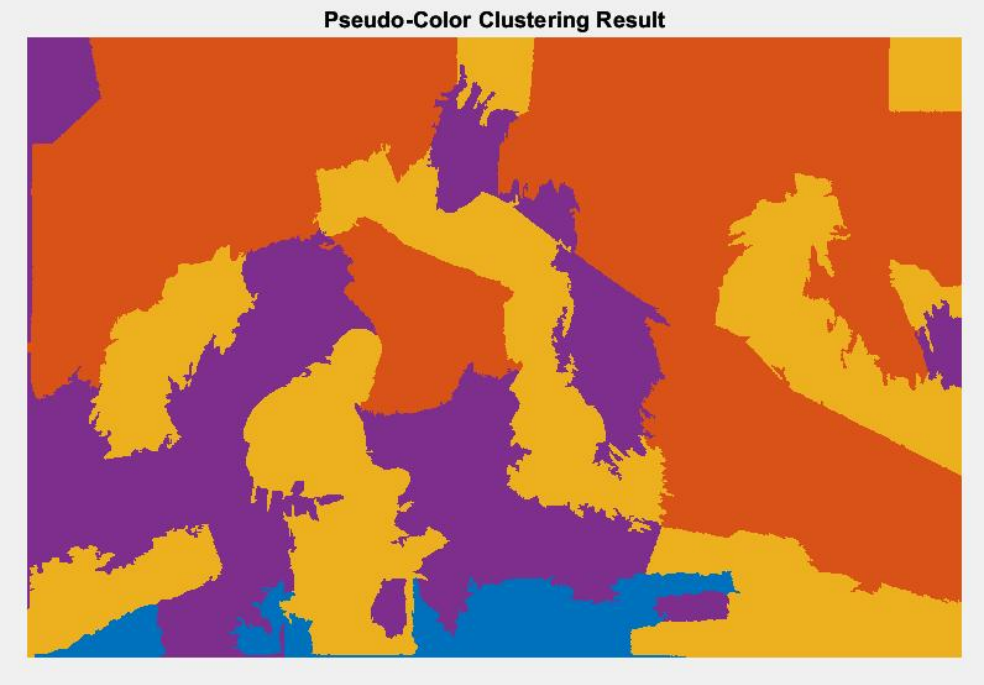


Figure 2.13: Pseudo-Color Image 3

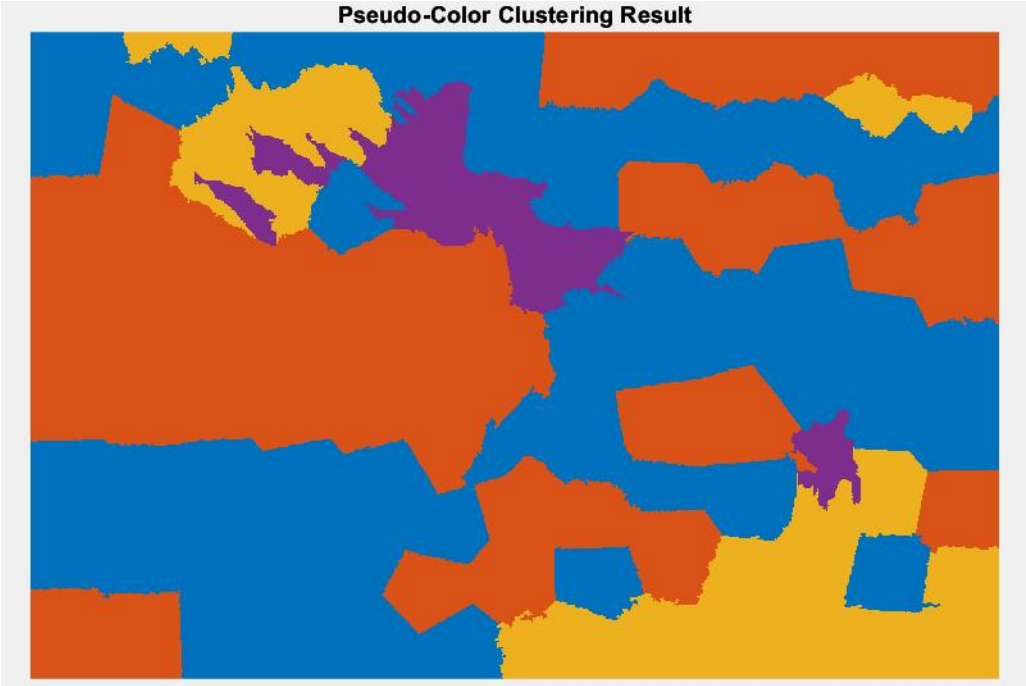


Figure 2.14: Pseudo-Color Image 4

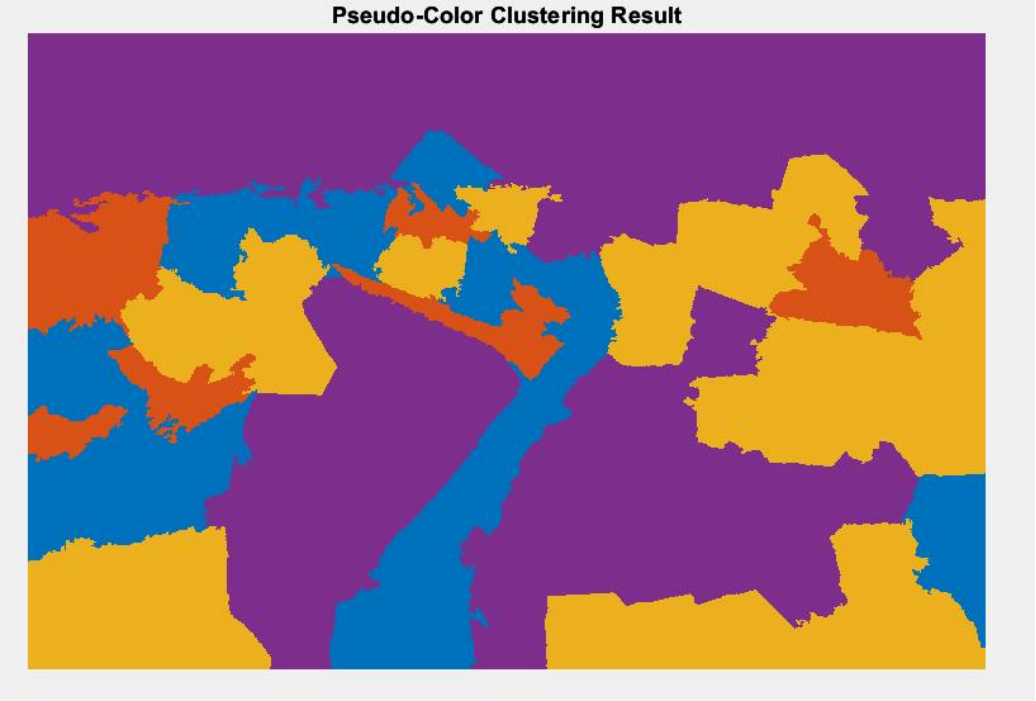


Figure 2.15: Pseudo-Color Image 5

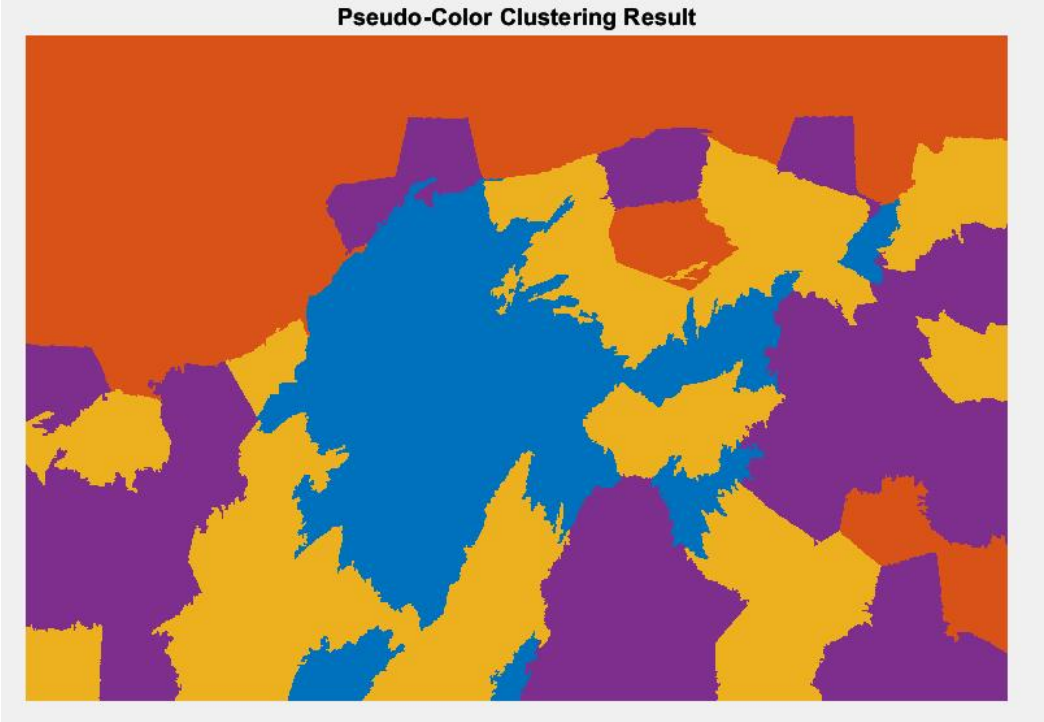


Figure 2.16: Pseudo-Color Image 6

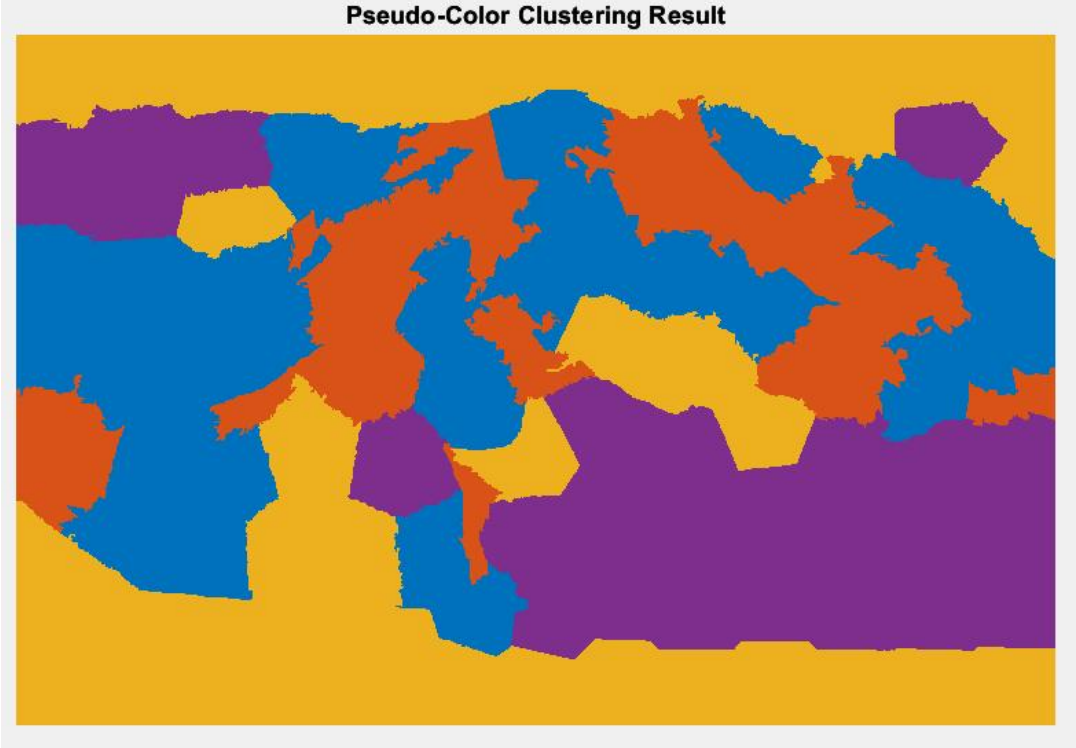


Figure 2.17: Pseudo-Color Image 7

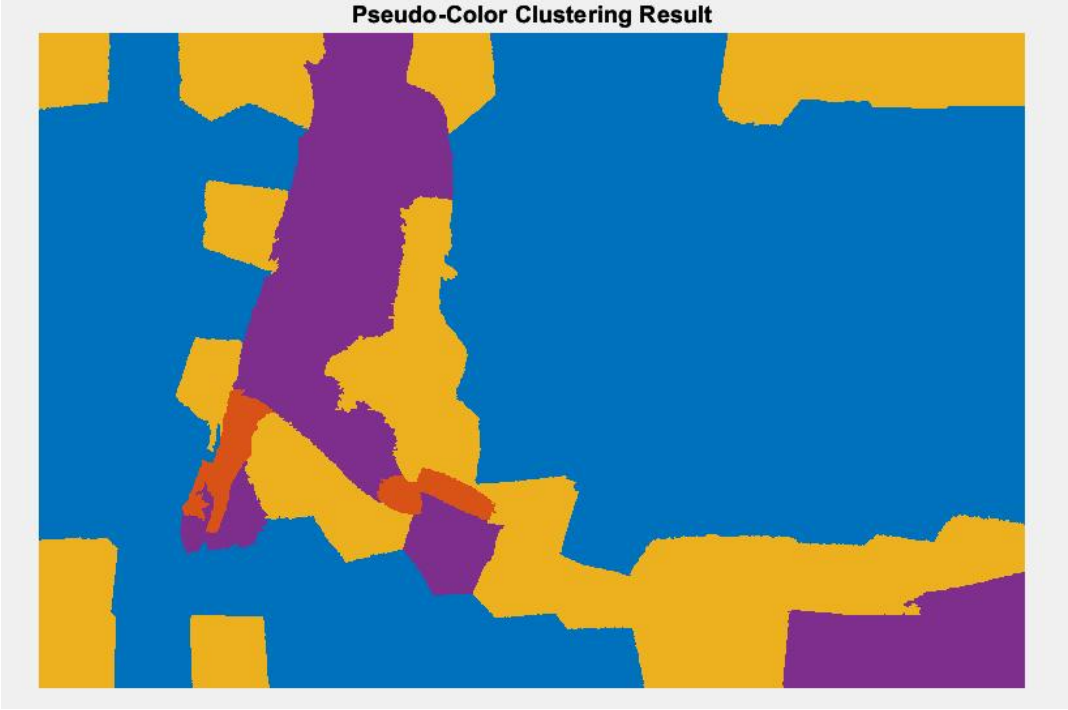


Figure 2.18: Pseudo-Color Image 8

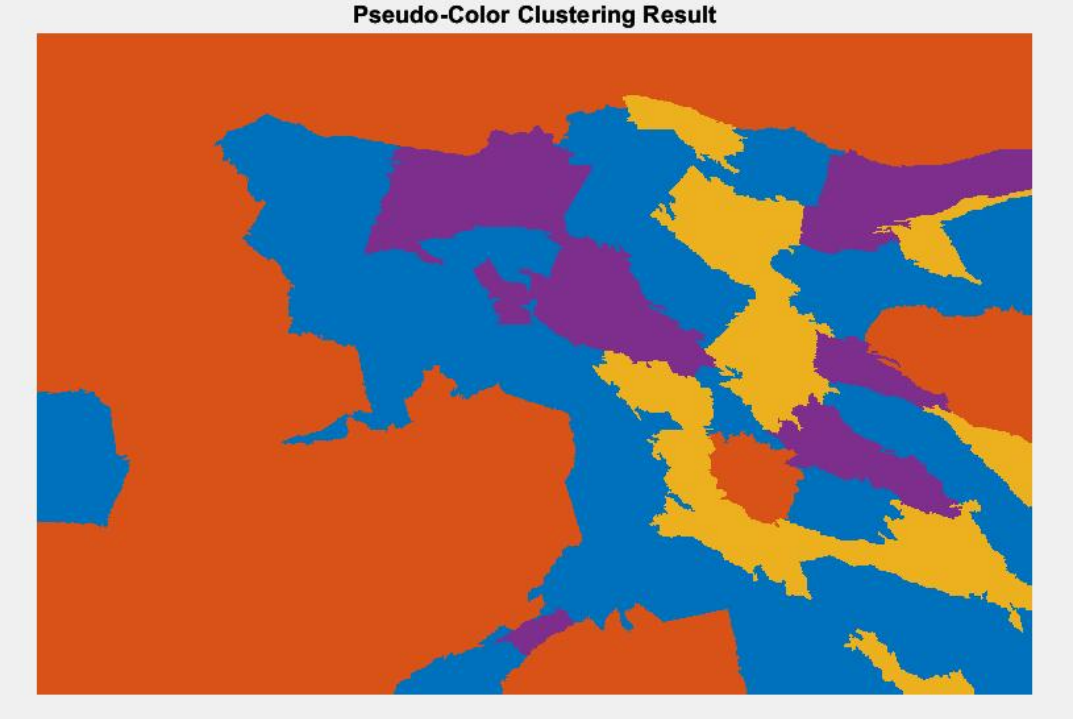


Figure 2.19: Pseudo-Color Image 9

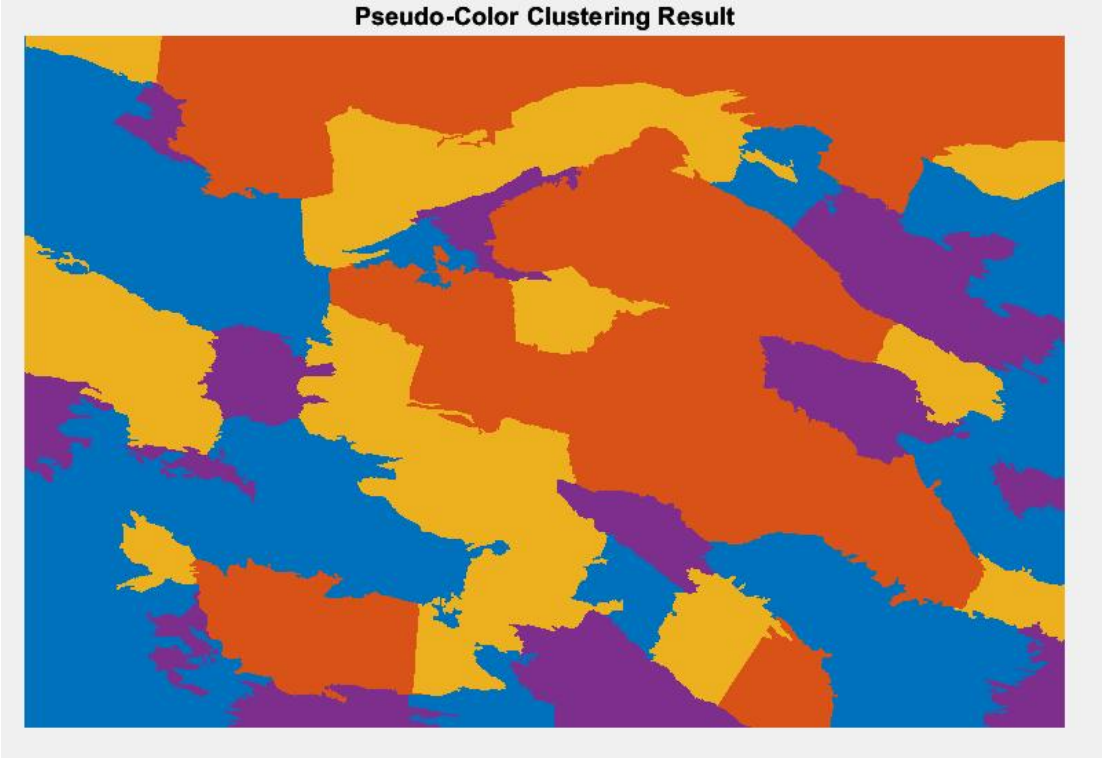


Figure 2.20: Pseudo-Color Image 10

References

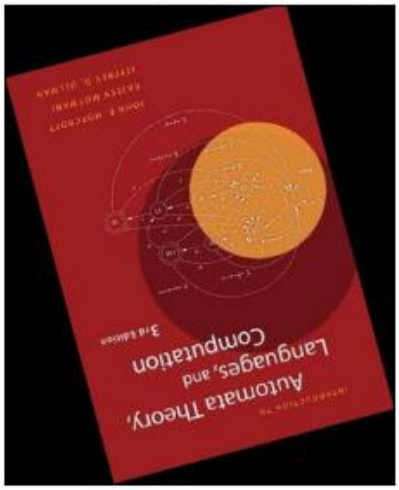
[1] “Canny Edge Detection,” OpenCV, https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html (accessed Nov. 30, 2024).

[2] GeeksforGeeks, "Line Detection using Python OpenCV HoughLine Method," GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/line-detection-python-opencv-houghline-method/. [Accessed: Dec. 1, 2024].

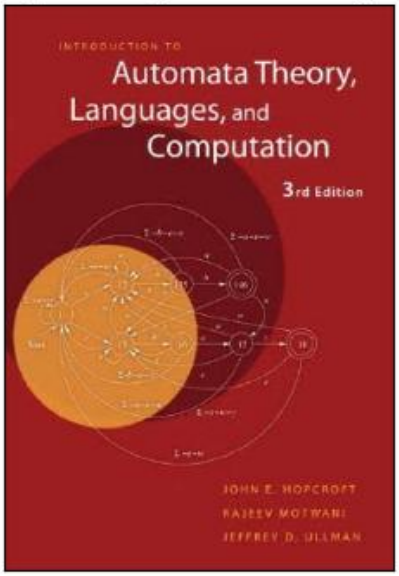
[3] P. Kovesi, "Peter's functions for computer vision," Peter Kovesi's Website, [Online]. Available: https://www.peterkovesi.com/matlabfns/. [Accessed: Dec. 1, 2024].

Appendix A - The matching results

Rotated Image: automata.jpeg

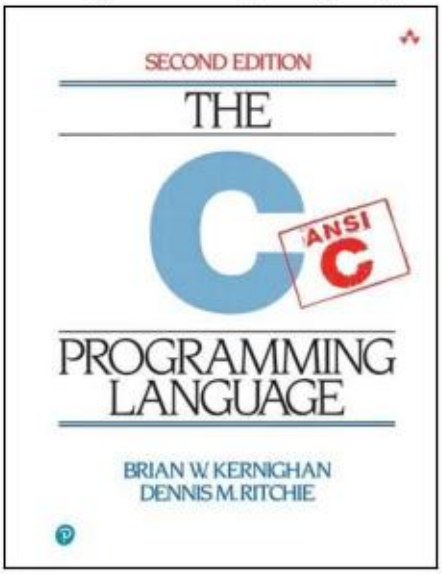


Original Image: automata.jpeg

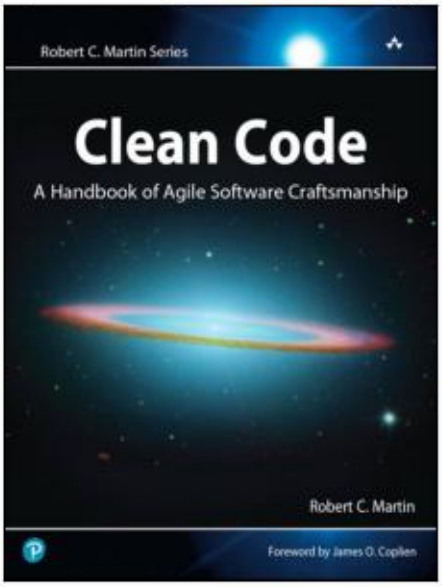


Rotated Image: c.jpeg

Original Image: c.jpeg

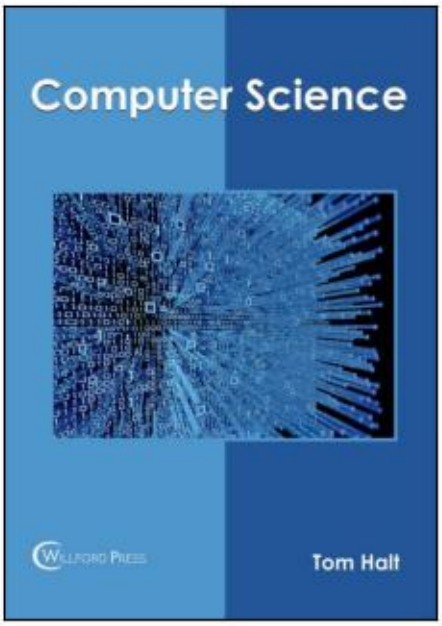


Original Image: cleancode.jpeg



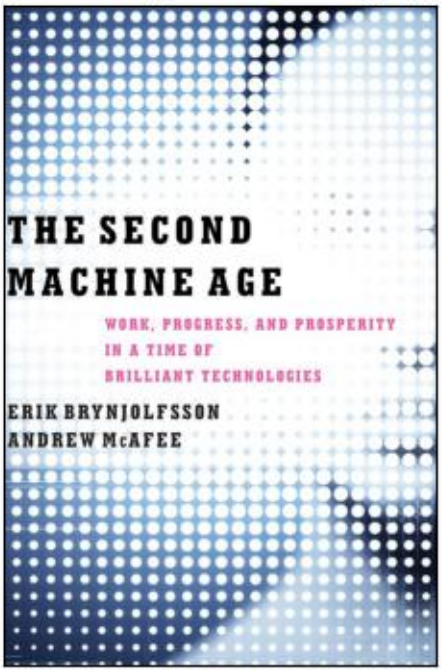
Rotated Image: cleancode.jpeg

Original Image: cs.jpeg



Rotated Image: cs.jpeg

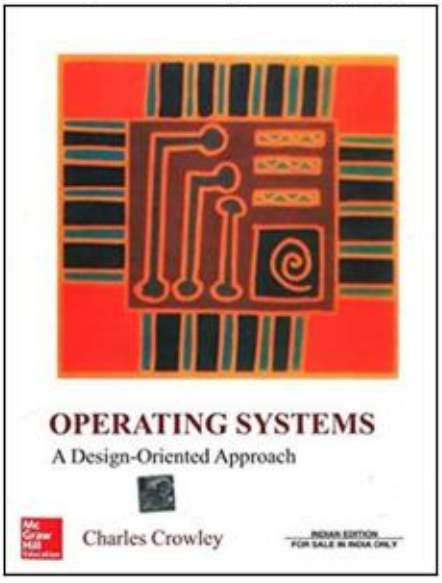
Original Image: machineage.jpeg



Rotated Image: cv.jpeg

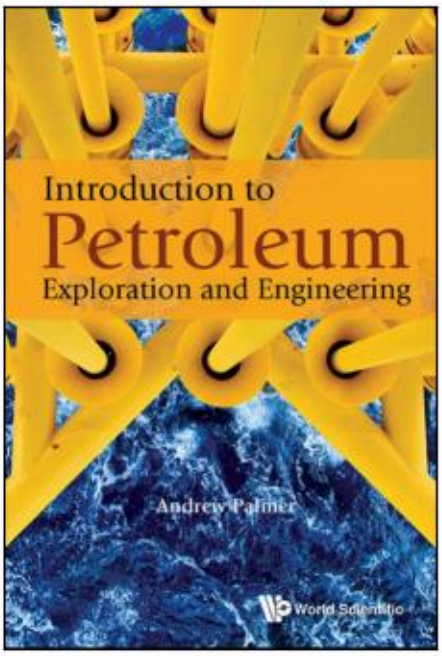
Rotated Image: fn.jpeg

Original Image: os.jpeg



Rotated Image: machineage.jpeg

Original Image: petrol.jpeg

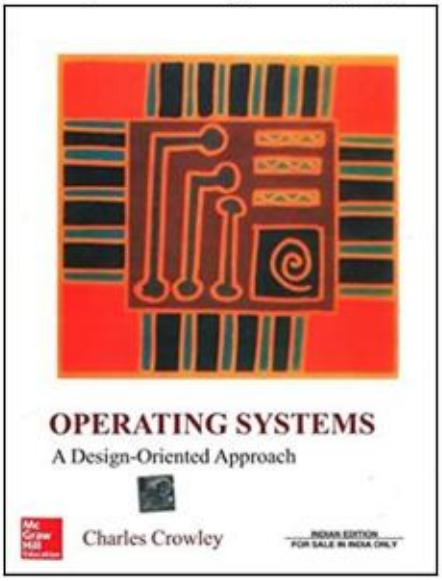


Rotated Image: noname.png

Original Image: noname.png

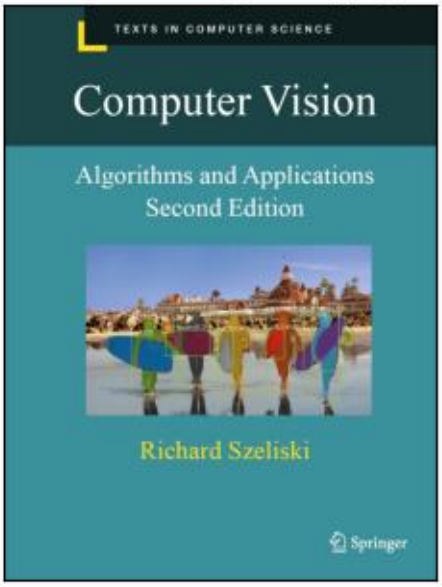
Rotated Image: os.jpeg

Original Image: os.jpeg

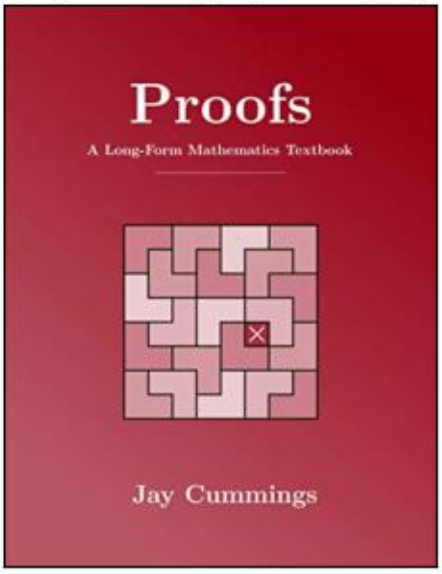


Rotated Image: petrol.jpeg

Original Image: cv.jpeg

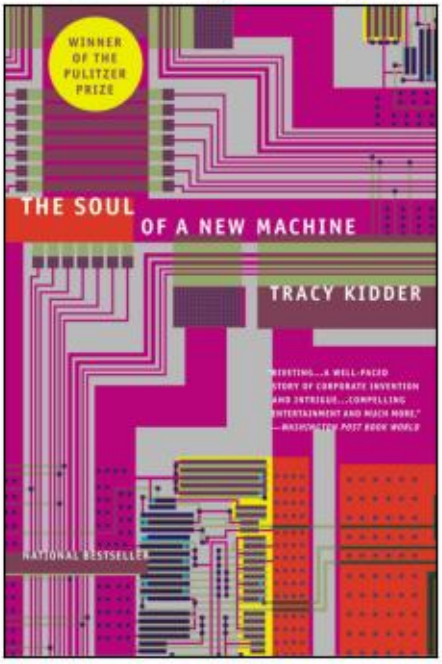


Original Image: proofs.jpeg

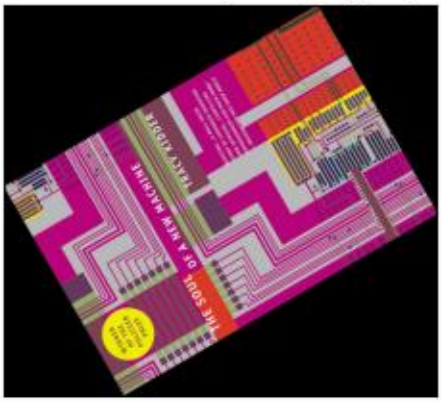


Rotated Image: proofs.jpeg

Original Image: soul.jpeg

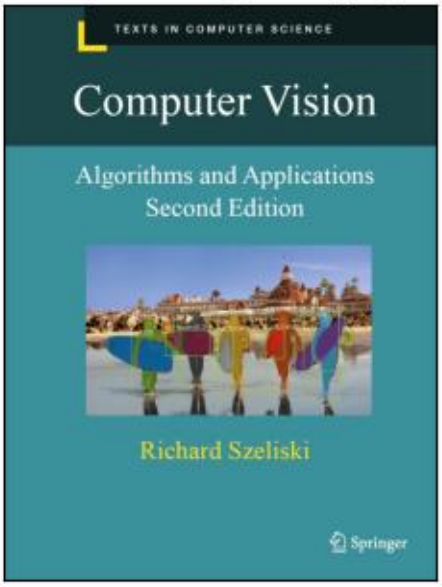


Rotated Image: soul.jpeg



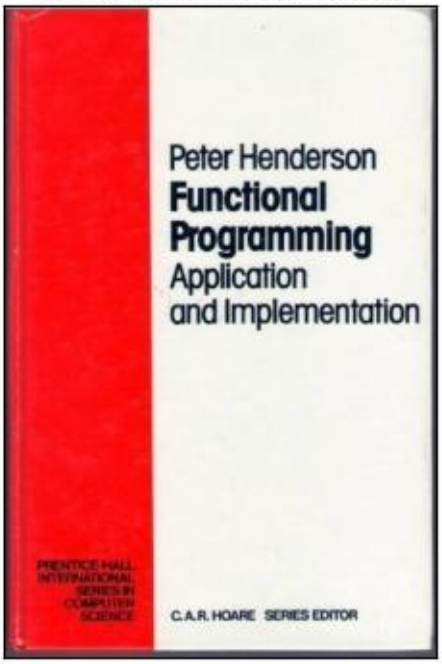
Rotated Image: strip.png

Original Image: cv.jpeg



Rotated Image: topology.jpeg

Original Image: fn.jpeg



Rotated Image: understory.jpeg

Original Image: understory.jpeg