Istanbul Technical University
Computer Engineering Department
BLG 453E - Computer Vision - Fall 24/25

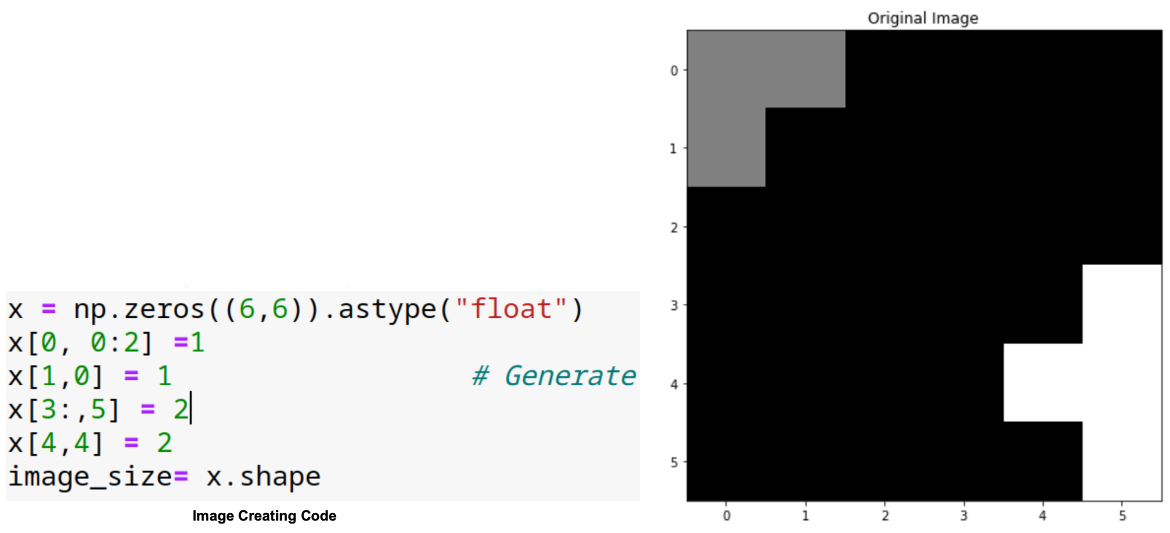
Due: 05.12.2024, 23.59 PM   Fall 24/25, Semester   Assigment 2

Notes

- You should do your own work! . Cheating is highly discouraged.
- You can use built-in Python, Numpy, and OpenCV functions until otherwise stated.
- You should write your codes in Python.
- Partial points will be given for incomplete solutions.
- Report creation is optional, but may be subject to bonus points if it is well-written. If you do not prepare a report, the codes will be expected to be well-commented.
- For your questions, do not hesitate to reach Res. Asst. Ziya Ata Yazıcı (yaziciz21@itu.edu.tr)and Res. Asst. Tuğçe Temel (temel21@itu.edu.tr).

Q1 (30 pts) Segment Image with Otsu's Algorithm

Consider an image with 2 objects and a total of 3 pixel values (use image creation code at Figure 1).



```
x = np.zeros((6,6)).astype("float")
x[0, 0:2] =1
x[1,0] = 1                        # Generate
x[3:,5] = 2
x[4,4] = 2
image_size= x.shape
```

Figure 1: Dummy image creating code and output image

Q1.1 (10 pts) Gaussian Noise

Add Gaussian noise to the image and you will get an image like the Figure 2.

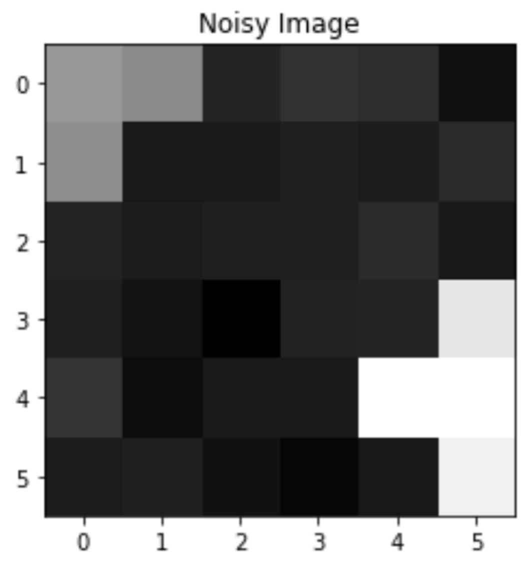


Figure 2: Noisly Image

Q1.2 (20 pts) Otsu's Algorithm
Implement Otsu's algorithm on this image to segment objects and test it for at least 2 different T. You can check this site for more details.

The implementation steps to Otsu:

- **Histogram Calculation (5 pts)**:
  Compute the histogram of the image. This represents the frequency of each pixel intensity level (e.g., 0–255 for an 8-bit grayscale image).
- **Probability Mass Function (PMF):**
  The Probability Density Function (PDF) is defined as:
  $$\text{PDF}[i] = \frac{h(i)}{\text{Total number of pixels}}$$
- Cumulative Distribution Function (CDF) (5 pts): The **Cumulative Distribution Function (CDF)** is the cumulative sum of the PDF up to intensity T:
  $$\text{CDF}[T] = \sum_{i=0}^{T} \text{PDF}[i]$$
- Mean and Variance Calculation
- Maximizing Between-Class Variance:
  The between-class variance $\sigma_B^2(T)$ is defined as:
  $$\sigma_B^2(T) = w_1(T) \cdot w_2(T) \cdot [\mu_1(T) - \mu_2(T)]^2$$
  where:
  - $w_1(T)$: Probability of Class 1 (background) = CDF[T]
  - $w_2(T)$: Probability of Class 2 (foreground) = $1 - w_1(T)$
  - $\mu_1(T)$: Mean intensity of Class 1
  - $\mu_2(T)$: Mean intensity of Class 2
- Optimal Threshold Selection: Iterate over all T (from 0 to 255), tracking the maximum $\sigma_B^2(T)$.
  The optimal threshold T^* is given by:
  $$T^* = \arg\max_T \sigma_B^2(T)$$

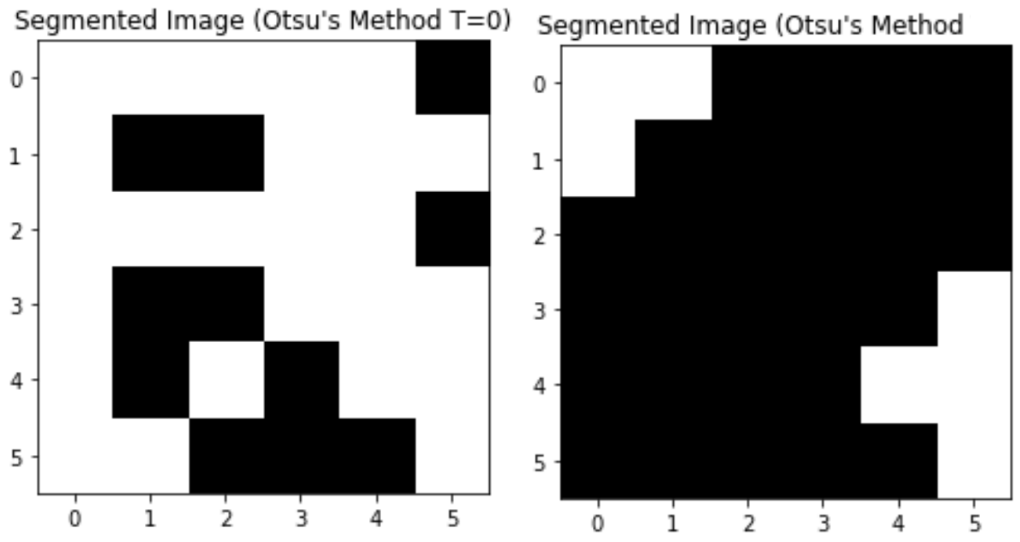


Figure 3: Segmented Image with different T

If implemented correctly, the output will be as follows Figures 3.

Q2 (20 pts) Image Noise Addition and Removal?

- Add salt-and-pepper noise and impulse noise to Lenna.png image, seperately.
- Apply an appropriate filter to remove the salt-and-pepper noise and impulse noise from the Lenna.png.

You are required to use Python and image processing libraries such as scikit-image, OpenCV, or NumPy to perform the task. You will also visualize the results using matplotlib.

Q3 (35 pts) How can read the book cover?

A student has taken a photo of a book cover at an angle (Figure 4), and the text appears distorted due to the perspective. Your task is to correct the perspective distortion and make the text appear as if viewed directly from the front. **Using the image "gogol.jpg" (available in the assignment folder)**.

Q3.1 (15 pts) Load image and Find corner

- Please firstly, load the image using the one of them OpenCV or Matplotlib library. Then detect the corners. (**Hint:** Use edge detection and contour detection methods to identify the boundary of the book cover.The shape of the book is rectangle so please be ensure the detected exactly 4 corner points.)
- Arrange the detected 4 corner points in a consistent order (e.g., top-left, top-right, bottom-right, bottom-left). Make sure the corner points are ordered in a consistent way otherwise you encounter the mirroring or rotation issues.

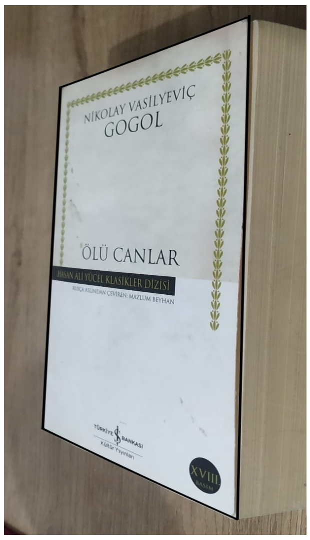
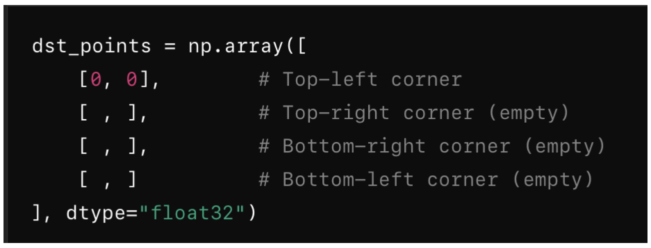


Figure 4: Side view of the book cover

Q3.2 (20 pts) Apply the Perspective Transformation

Before the transformation create an array list to destination points just like Figure 2. Top left corner coordinates are given for example, please fill the top-right, bottom-right and bottom-left according to image size.



```
dst_points = np.array([
    [0, 0],         # Top-left corner
    [ , ],          # Top-right corner (empty)
    [ , ],          # Bottom-right corner (empty)
    [ , ]           # Bottom-left corner (empty)
], dtype="float32")
```

Figure 5: Destination points' array

Use OpenCV's cv2.getPerspectiveTransform to calculating the perspective matrix and use cv2.warpPerspective functions for applying this perspective warp to the image. If you followed all the steps correctly, your output will look like Figure 6.

Q4 (25 pts) Understanding and Implementing Generative Adversarial Networks (GANs)

In this quesrion, you will learn about Generative Adversarial Networks (GANs) and implement a simple GAN model from scratch. The goal is to generate synthetic images that resemble a real dataset using a neural network. You will explore the components of GANs, their architecture, and how the two networks (Generator and Discriminator) compete to improve each other's performance.

Tasks: Understanding the GAN Architecture: Explain the basic concept behind Generative Adversarial Networks (GANs).Describe the roles of the two main components of a GAN:

- The Generator: What is its role, and how does it generate data ?
- The Discriminator: What is its role, and how does it classify data ?

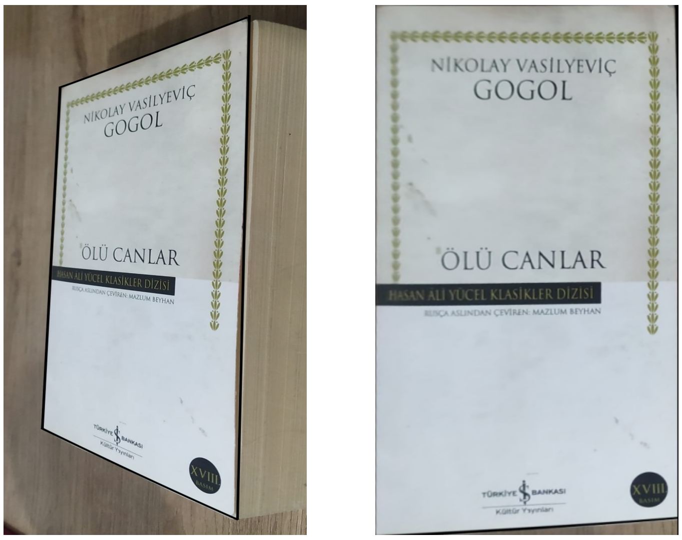

NİKOLAY VASİLYEVİÇ
GOGOL
ÖLÜ CANLAR
HASAN ALİ YÜCEL KLASİKLER DİZİSİ
RUSÇA ASLINDAN ÇEVİREN: MAZLUM BEYHAN
TÜRKİYE BANKASI
Kültür Yayınları
XVIII
NİKOLAY VASİLYEVİÇ
GOGOL
ÖLÜ CANLAR
HASAN ALİ YÜCEL KLASİKLER DİZİSİ
RUSÇA ASLINDAN ÇEVİREN: MAZLUM BEYHAN
TÜRKİYE BANKASI
Kültür Yayınları
XVIII

Figure 6: Original image (Right side view) and After applying Perspective Transformation (Front view)

- Discuss the adversarial training process and how the Generator and Discriminator improve through their competition.

GAN Implementation: Implement a simple GAN using TensorFlow or PyTorch. Use a simple dataset like **CIFAR-10** (small color images) for training your GAN.
Your model should consist of:

- A Generator model that takes random noise as input and generates synthetic images.
- A Discriminator model that classifies whether an image is real (from the dataset) or fake (generated by the Generator).
- A Loss Function for both the Generator and the Discriminator that helps guide their training.

Train your GAN on the selected dataset. Visualize the generated images at different stages of training (e.g., before training, halfway through training, and after the model has fully trained). Analyze the quality of the generated images. Do they resemble the real images from the dataset? How do they evolve over time?