CSE166_FA23_assignment_1

October 4, 2023

1 CSE 166 Image Processing, Fall 2023 - Assignment 1

Instructor: Ben Ochoa

Assignment due: Wed, Oct 11, 11:59 PM

Name:

PID:

1.1 Prior knowledge & certification of commencement of academic activity

In every course at UC San Diego, per the US Department of Education, we are now required to certify whether students have commenced academic activity for a class to be counted towards eligibility for Title IV federal financial aid. This certification must be completed during the first two weeks of instruction.

For CSE 166, this requirement will be fulfilled via an ungraded prior knowledge quiz, which will assist the instructional team by providing information about your background coming into the course. In Canvas, go to the CSE 166 course and navigate to Quizzes. Then, click on the "First Day Survey: Prior Knowledge #FinAid"

1.2 Instructions

1. All solutions to the problems below must be written in this notebook. Programming aspects of the assignment must be completed using Python (preferably 3.10). Please check the **README.md** for setup of the virtual environment.
2. This assignment must be completed **individually**. For more details, please review the Academic Integrity Policy and Collaboration Policy on Canvas.
3. The packages you need to complete the assignment will be indicated in the problem descriptions.You are not allowed to use the packages that directly solve the problems. Feel free to ask the instructor and TAs if you are unsure about the packages to use.
4. It is highly recommended that you begin working on this assignment early.
5. After finishing the assignment in the notebook, please export the notebook as a PDF and a python source file. You need to **submit 3 files: the Notebook, the PDF and the python file** (i.e. the `.ipynb`, the `.pdf` and the `.py` files) on Gradescope.

- To convert the notebook to PDF, you can choose one way below:
  - You may first export the notebook as HTML, and then print the web page as PDF

    - e.g., in Chrome: File → Save and Export Notebook as → "HTML"; or in VS-code: Open the Command Palette by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (macOS), search for Jupyter: Export to HTML
    - Open the saved web page and right click → Print... → Choose "Destination: Save as PDF" and click "Save")
  - If you have XeTex installed on your machine, you may directly export the notebook as PDF: e.g., in Chrome, File → Save and Export Notebook as → "PDF"
  - You may use nbconvert to convert the ipynb file to pdf using the following command `jupyter nbconvert --allow-chromium-download --to webpdf filename.ipynb`
- To convert the notebook to python file, you can choose one way below:
  - You may directly export the notebook as py: e.g., in Chrome, File → Save and Export Notebook as → "Executable script"; or in VScode: Open the Command Palette and search for Jupyter: Export to Python Script
  - You may use nbconvert to convert the ipynb file to python file using the following command `jupyter nbconvert --to script filename.ipynb`

6.. Please make sure the content in each cell (e.g. code, output images, printed results, etc.) are clearly visible and are not cut-out or partially cropped in your final PDF file.

7. While submitting on gradescope, please make sure to assign the relevant pages in your PDF submission for each problem.

Late Policy: Assignments submitted late will receive a 15% grade reduction for each 12 hours late (i.e., 30% per day). Assignments will not be accepted 72 hours after the due date. If you require an extension (for personal reasons only) to a due date, you must request one as far in advance as possible. Extensions requested close to or after the due date will only be granted for clear emergencies or clearly unforeseeable circumstances.

2 Problem 1: 2D transformation matrices (5 points)

Given the 2D transformation matrices

$\mathtt{H}_{\mathrm{t}} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$ and $\mathtt{H}_{\mathrm{R}} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ show that $\mathtt{H} = \mathtt{H}_{\mathrm{t}}^{-1}\mathtt{H}_{\mathrm{R}}\mathtt{H}_{\mathrm{t}}$ is a 2D Euclidean transformation matrix.

Your answer here:

3 Problem 2: Programming: Basic manipulations of matrices (5 points)

Import necessary packages for Problem 2 and 3.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
import math
```

Define two matrices $\texttt{A} = \begin{bmatrix} 1 & 3 & 5 & 1 \\ 4 & 7 & 4 & 6 \\ 6 & 1 & 3 & 0 \\ 6 & 2 & 7 & 0 \end{bmatrix}$ and $\texttt{B} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$ with numpy array.

1. Calculate the point-wise multiplication of A and B. Set the result to C and print C.

[ ]:
```
# your code here
```

2. Calculate the matrix multiplication AB. Set the result to D and print D.

[ ]:
```
# your code here
```

3. Calculate the inner product of the 1st column of C and 4th row of D. Print your result.

[ ]:
```
# your code here
```

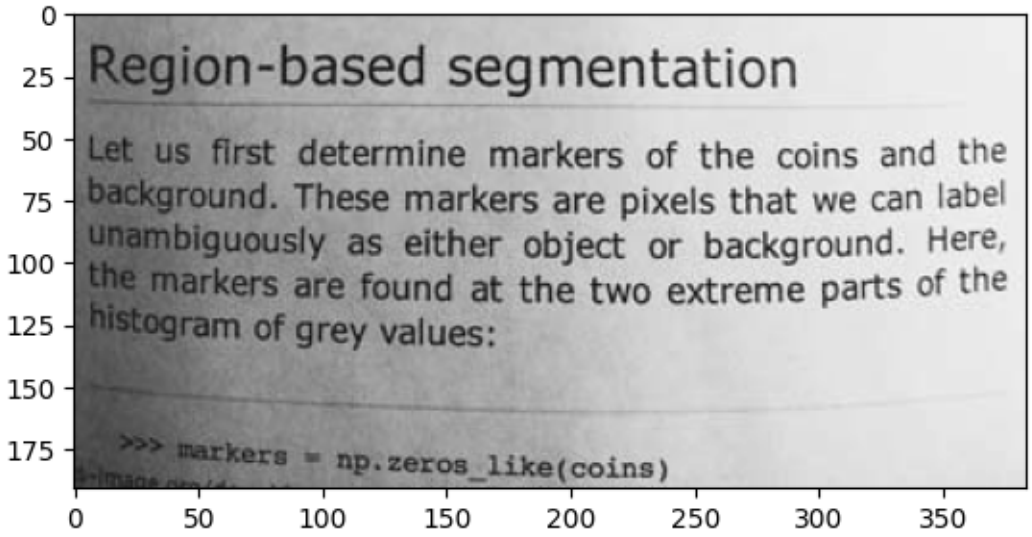
4. Find the minimum values and their corresponding row and column indices (one-based) in matrix D by using numpy functions. If there are multiple min values, you must list all their indices. Print your result.

[ ]:
```
# your code here
```

4 Problem 3: Programming: Transform images (30 points)

In this problem, you are provided with a text image. This problem consists of 3 parts. In the first part, you are expected to write a function to compute a 2D transformation matrix for rotating this image about its center by a given angle. In the second and third parts, you are expected to rotate the image using two different interpolation methods discussed in class - the nearest neighbor interpolation and the linear interpolation.

[ ]:
```
# Read and display image
img = data.page()
plt.imshow(img, cmap='gray', vmin=0, vmax=255)
plt.show()
print('The image size is '+ str(img.shape))
```




```
The image size is (191, 384)
```

4.1 Part 1: 2D transformation matrix (5 points)

1. Complete the function `get_transformation_matrix` that computes a 2D transformation matrix to rotate an image about its center. The function inputs are image width, image height, and rotation angle. The output of the function should be the 2D transformation matrix.

```
[ ]: # function that calculates the 2D transformation matrix for rotating an image␣
      ↪about its center
     def get_transformation_matrix(img_ht, img_wt, rot):
       """
       input:
       img_ht: image height in pixels
       img_wt: image width in pixels
       rot: rotation angle in radians

       output:
       h: 2D transformation matrix
       """
       # your code here
       h = []

       return h
```

2. Compute the output of the function `get_transformation_matrix` for the image "img" we defined above with an rotation angle = $\frac{\pi}{5}$. Print the numerical result.

[ ]:
```
"""
Call the function `get_transformation_matrix` for img, and rotation angle = pi/5.
Print the result.
"""
# your code here
```

3. What would an angle of $-\frac{5\pi}{6}$ correspond to in degrees? Give your answer in the range $[0, 360)$.

Your answer here:

4.2 Part 2: Image transformation (Nearest Neighbor interpolation) (10 points)

1. Complete the function `rotate_nearest_neighbor` that takes an image as input and rotates the image about its center using the nearest neighbor interpolation method. The function inputs are an image and a 2D transformation matrix. The function should output and return the transformed image. The function must set those pixels to black in the output image that inversely map to pixels outside of the input image boundaries.

Note: The output image must be of the same size as the input image.

[ ]:
```
# function that rotates image and applies nearest neighbor interpolation
def rotate_nearest_neighbor(img, h):
  """
  img: source image
  h: 2D transformation matrix

  returns:
  rot_img: image after rotation
  """
  # your code here
  rot_img = []

  return rot_img
```

2. Using the `rotate_nearest_neighbor` function, rotate the original image about its center for rotation angles $\frac{\pi}{5}$, $\frac{\pi}{3}$, $\frac{\pi}{2}$, and $\frac{5\pi}{6}$. You must call the function `get_transformation_matrix` to calculate the 2D transformation matrix for each of these rotation angles. Display transformed images for each rotation angles, and clearly label which output images correspond to which rotation angles. Set the value range to [0, 255] on all display calls.

[ ]:
```
"""
Call the `rotate_nearest_neighbor` function for rotating images using nearest␣
↪neighbor interpolation method.
Display the original image and rotated images for rotation angles pi/5, pi/3,␣
↪pi/2 and 5*pi/6.
"""
# your code here
```

4.3 Part 3: Image transformation (Linear interpolation) (15 points)

1. Complete the function `rotate_linear` that takes an image as input and rotates the image about its center using the linear interpolation method. The function inputs are an image and a 2D transformation matrix. The function should return the transformed image. The function must set those pixels to black in the output image that inversely map to pixels outside of the input image boundaries.

Note: The output image must be of the same size as the input image.

[ ]:
```
# function that rotates image and applies linear interpolation
def rotate_linear(img, h):
  """
  img: source image
  h: 2D transformation matrix

  returns:
  rot_img: image after rotation
  """
  # your code here
  rot_img = []

  return rot_img
```

2. Using the `rotate_linear` function, rotate the original image about its center for rotation angles $\frac{\pi}{5}$, $\frac{\pi}{3}$, $\frac{\pi}{2}$, and $\frac{5\pi}{6}$. You must call the function `get_transformation_matrix` to calculate 2D transformation matrix for each of these rotation angles. Display transformed images for each rotation angles, and clearly label which output images correspond to which rotation angles. Set the value range to [0, 255] on all display calls.

[ ]:
```
"""
Call the `rotate_linear` function for rotating images using linear␣
 ↪interpolation method.
Display the original image and rotated images for rotation angles pi/5, pi/3,␣
 ↪pi/2 and 5*pi/6.
"""
# your code here
```

3. Briefly discuss the qualitative differences between the results with the nearest neighbor interpolation and the linear interpolation.

Your answer here: