CSE166_WI23_assignment_1

January 11, 2023

1 CSE 166 Image Processing, Winter 2023 - Assignment 1

- Instructor: Ben Ochoa
- Due: Wednesday, January 18, 2023, 11:59 PM

1.1 Prior knowledge + certification of commencement of academic activity

Beginning this past summer, 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

Please answer the questions below using Python in the attached Jupyter notebook and follow the guidelines below:

- This assignment must be completed **individually**. For more details, please review the Academic Integrity Policy and Collaboration Policy on Canvas.
- All the solutions must be written in the Jupyter notebook only.
- After finishing the assignment in the notebook, please export the notebook as a PDF and **submit both the Notebook and the PDF** (i.e. the `.ipynb` and the `.pdf` files) on Gradescope. While submitting the PDF on gradescope -
  - Make sure that the outputs generated are clearly visible and are not cut-off or partially cropped in the final PDF.
  - Make sure to assign the relevant pages in your PDF submission for each problem.
  - Do not export the jupyter notebook as a single page. Please see the detailed submission instructions at the end of this file.
- You may use basic algebra packages (e.g. `NumPy`, `SciPy`, etc) but you are not allowed to use the packages that directly solve the problems. Feel free to ask the instructor and the teaching assistants if you are unsure about the packages to use.
- It is highly recommended that you begin working on this assignment early.

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

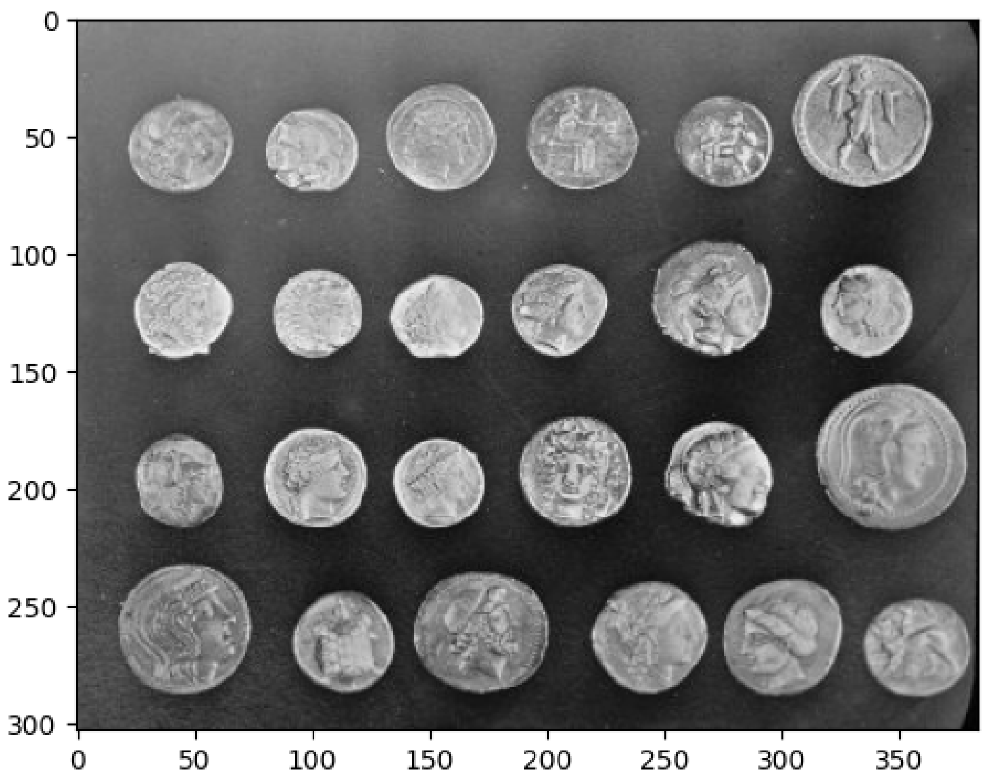
3 Problem 2: Programming: Transform images (30 points)

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

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

```
[ ]: # Read and display image
     img = data.coins()
     plt.imshow(img, cmap='gray')
```

```
[ ]: <matplotlib.image.AxesImage at 0x190b98ce0a0>
```



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

1. Develop a Python function called `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 function should output and return the 2D transformation matrix.

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

  returns:
  h: 2D transformation matrix
  """
  # your code here

  return h
```

2. Compute the output of the function `get_transformation_matrix` for image width = 640, image height = 480, rotation angle = $\frac{\pi}{3}$. Print the numerical results.

```
[ ]: """
     Call the function `get_transformation_matrix` for image width = 640, image␣
      ↪height = 480, and rotation angle = pi/3.
     Print the result.
     """
     # your code here
```

3. Additionally, if a rotation angle of $\frac{\pi}{6}$ rotates images by 30 degrees, and $\frac{\pi}{4}$ rotates images by 45 degrees, what would an angle of $-\frac{5\pi}{6}$ correspond to in degrees? Give your answer in the range $[0, 360)$.

Your answer here

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

1. Develop a Python function called `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

         return rot_img
```

2. Using the `rotate_nearest_neighbor` function, rotate the original image about its center for rotation angles $\frac{\pi}{3}$, $\frac{\pi}{2}$, $\frac{5\pi}{6}$, and $\frac{3\pi}{2}$. You must call the function `get_transformation_matrix` to calculate the 2D transformation matrix for each of these rotation angles. Display the input image and transformed images for each rotation angles, and clearly label which output images correspond to which rotation angles.

```
[ ]: """
     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/3, pi/2,␣
 ↪5*pi/6 and 3*pi/2.
"""
# your code here
```

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

1. Develop a Python function called `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 images 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

    return rot_img
```

2. Using the `rotate_linear` function, rotate the original image about its center for rotation angles $\frac{\pi}{3}$, $\frac{\pi}{2}$, $\frac{5\pi}{6}$, and $\frac{3\pi}{2}$. You must call the function `get_transformation_matrix` to calculate 2D transformation matrix for each of these rotation angles. Display the input image and transformed images for each rotation angles, and clearly label which output images correspond to which rotation angles.

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

3. Briefly discuss the qualitative differences between these results and those obtained using nearest neighbor interpolation.

Your answer here

4 Submission Instructions

1. Remember to submit **both** the Jupyter notebook file (`.ipynb`) and the PDF version (`.pdf`) of this notebook to Gradescope.
2. 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.
3. While submitting on gradescope, please make sure to assign the relevant pages in your PDF submission for each problem.
4. Do not export the jupyter notebook as a single page.

To convert the notebook to PDF, you can choose one way below:

1. You can print the web page and save as PDF (e.g. Chrome: Right click the web page → Print... → Choose "Destination: Save as PDF" and click "Save").
2. You can find the export option in the header: File → Download as → "PDF via LaTeX"
3. You can use nbconvert (https://nbconvert.readthedocs.io/en/latest/install.html) to convert the ipynb file to pdf using the following command

```
jupyter nbconvert --allow-chromium-download --to webpdf <ipynb filename>
```