CSE 166: Image Processing, Spring 2022 – Assignment 1

Instructor: Ben Ochoa

- Due On: **Wednesday, April 6, 2022, 11:59 PM (Pacific Time)**.

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 follow the Academic Integrity Policy and Collaboration Policy on Canvas.
- All the solutions must be written in this Jupyter notebook.
- 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.
- 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.

Problem 1: 2D transformation matrices (5 points)

Given the 2D transformation matrices

$$H_t = \begin{vmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{vmatrix} \text{ and } H_R = \begin{vmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

show that $H = H_t^{-1} H_R H_t$ is a 2D Euclidean transformation matrix.

your answer here

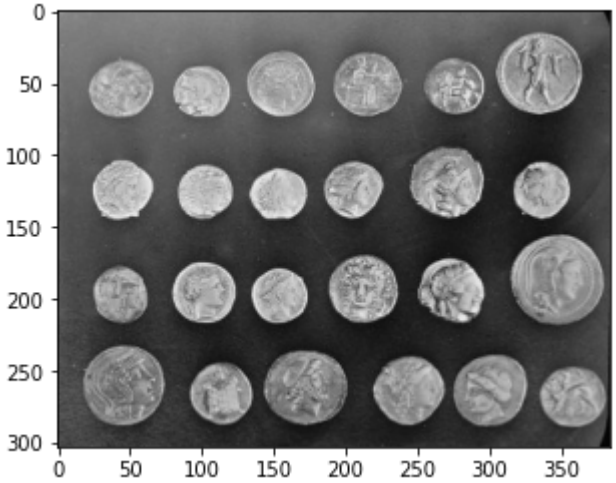
Problem 2: Programming: Transform images (30 points)

In this problem, you are provided with a sample coins image. You are expected to initially compute a 2D transformation matrix for rotating this image. Once you have the transformation matrix, 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 0x7f07160c5a90>
```



Part 1: 2D transformation matrix (5 points)

1. Develop a Python function called get transformation matrix that calculates the 2D transformation matrix that rotates an image about its center. The function inputs are image

   width, image height, and rotation angle. The function output is the 2D transformation matrix.
2. Compute and report the output of the function get transformation matrix for the rotation angle = π/6

```
# function that calculates the 2D transformation matrix for rotating an image ab
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
```

```
"""
Call the get_transformation_matrix matrix for the rotation angle = π/6. Print th
"""
# your code here

```

```
'\nCall the get_transformation_matrix matrix for the clockwise rotation angle
```

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

1. Develop a Python function called rotate_nn that rotates the image about the center with the nearest neighbor interpolation method. The function inputs are an image and a 2D transformation matrix. The function output is the transformed image. The function must set pixels to black in the output image that inversely map to pixels outside of the input image boundaries.
2. Display the rotated images for rotation angles π/6, π/2, 2π/3 and 3π/2;

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

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

  returns:
  rot_img: image after rotation
  """
  # your code here

  return rot_img
```

```
"""
Call the rotate_nn function for rotating images using nearest neighbor interpola
Display the rotated images for rotation angles π/6, π/2, 2π/3 and 3π/2;
"""
# your code here
```

```
'\nCall the rotate_nn function for rotating images using nearest neighbor int
erpolation.\nDisplay the rotated images for rotation angles π/6, π/2, 2π/3 an
d 3π/2:\n'
```

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

1. Develop a Python function called rotate_linear that rotates the image about the center with the linear interpolation method. The function inputs are an image and a 2D transformation matrix. The function output is the transformed image. The function must set pixels to black in the output image that inversely map to pixels outside of the input image boundaries.
2. Display the rotated images for rotation angles π/6, π/2, 2π/3 and 3π/2;
3. Briefly discuss the qualitative differences between these results and those obtained using nearest neighbor interpolation.

Note: The size of the output image must be 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
```

```
"""
Call the rotate_nn function for rotating images using linear interpolation.
Display the rotated images for rotation angles π/6, π/2, 2π/3 and 3π/2;
"""
# your code here
```

Briefly discuss the qualitative differences between these results and those

your answer here

```
1
```