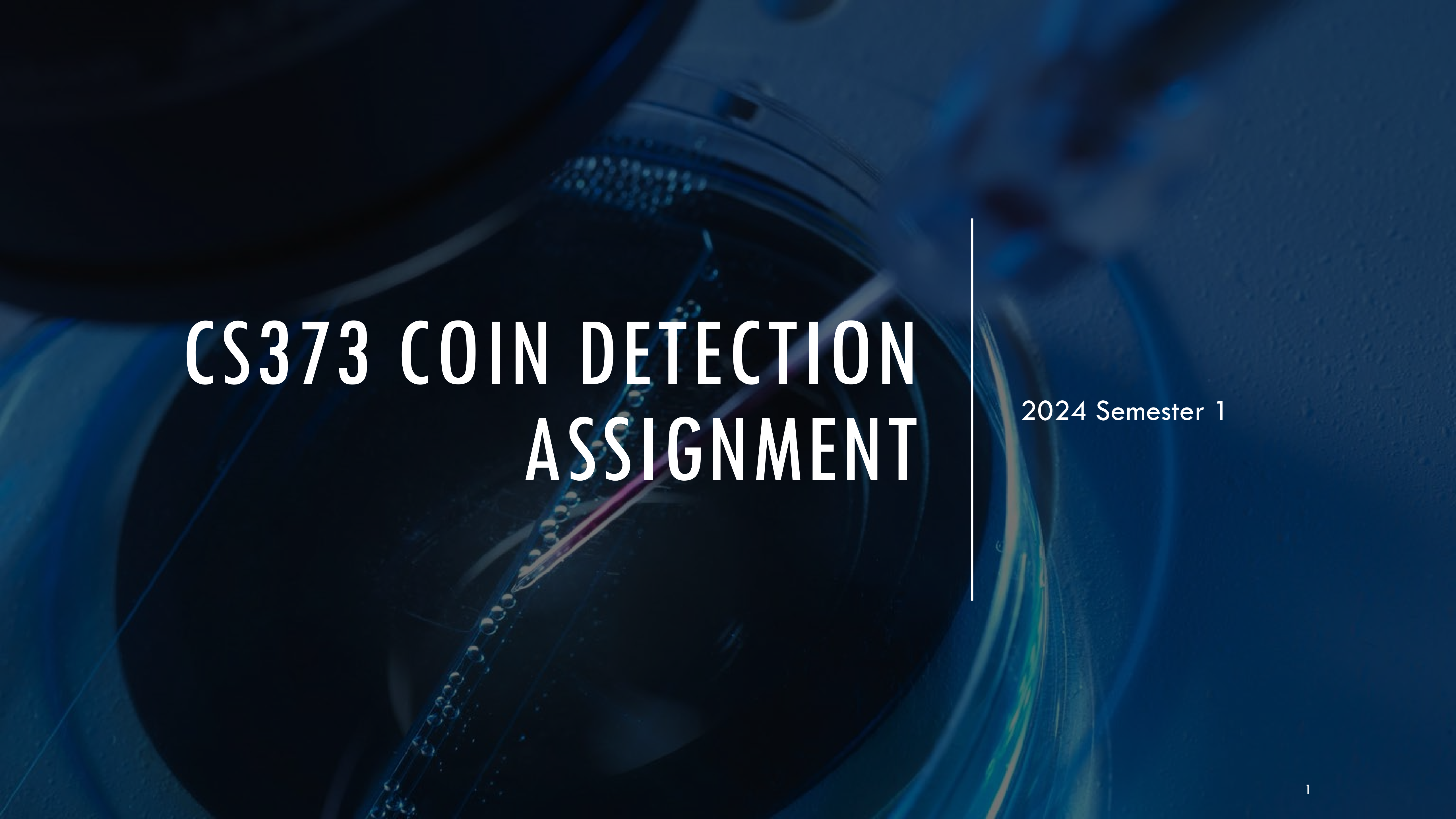


CS373 COIN DETECTION ASSIGNMENT

2024 Semester 1

Deadline: 3rd June 2024, 23:59pm

- In this assignment, you will write a Python code pipeline to automatically detect all the coins in the given images. This is an individual assignment, so every student has to submit this assignment! This assignment is worth 15 marks.

- We have provided you with 6 images for testing your pipeline (you can find the images in the 'Images/easy' folder).
  - Your pipeline should be able to detect all the coins in the image labelled with easy-level. This will reward you with up to 10 marks.
  - For extension (up to 5 marks), try images labelled as hard-level images in the "Images/hard" folder.
  - Write a short reflective report about your extension. (Using Latex/Word)

SUBMISSION

Please upload your submission as a zipped file of the assignment folder to the UoA Assignment Dropbox by following this link: https://canvas.auckland.ac.nz/courses/103807/assignments/383790

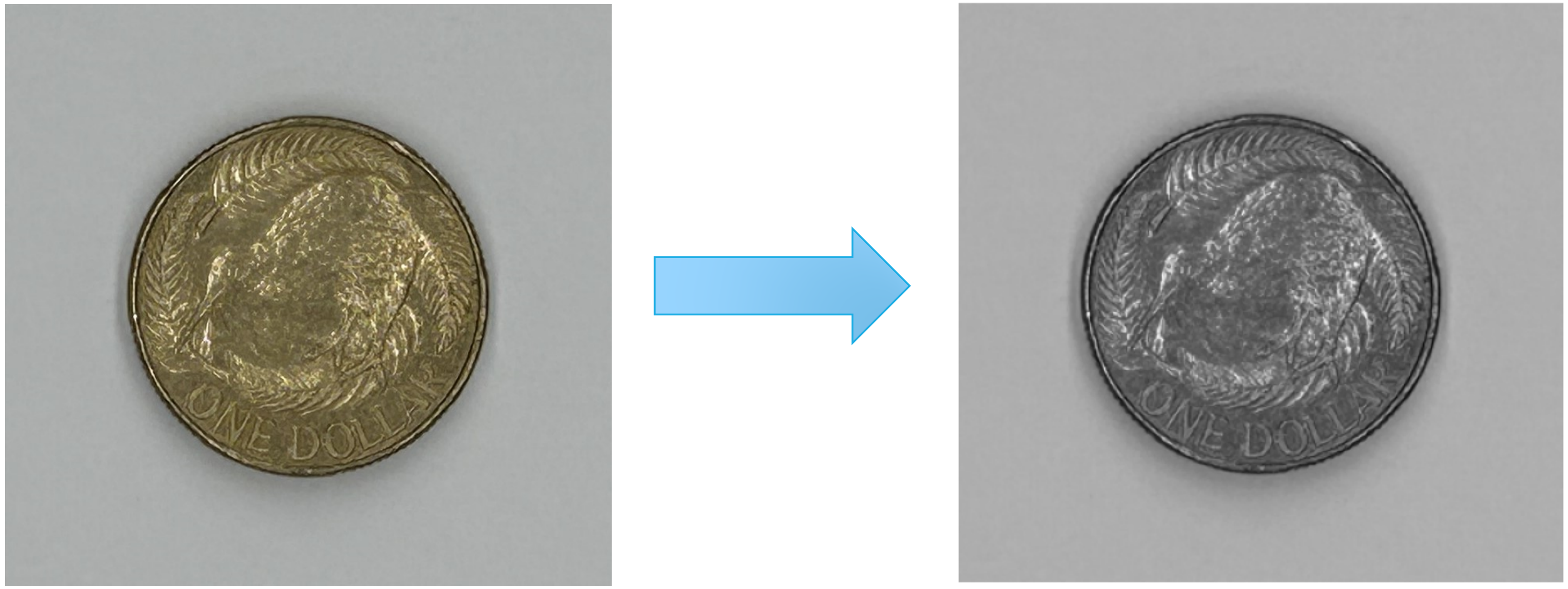
- Don't put any virtual environment (venv) folders into this zip file, it just adds to the size, and we will have our own testing environment.
- Your code for executing the main coin detection algorithm has to be located in the provided "CS373_coin_detection.py" file!
- You can either put all of your code into that file, or use a modular structure with additional files (that, of course, have to be submitted in the zip file). However, we will only execute the "CS373_coin_detection.py" file to see if your code works for the main component!
- The main component of the assignment ("CS373_coin_detection.py") must not use any non-built-in Python packages (e.g., PIL, OpenCV, NumPy, etc.) except for Matplotlib. Ensure your IDE hasn't added any of these packages to your imports.
- For the extensions, please create a new Python source file called '*CS373_coin_detection_extension.py*'; this will ensure your extension part doesn't mix up with the main component of the assignment. Remember, your algorithm has to pass the main component first!
- Including a short PDF report about your extension.
- Important: Use a lab computer to test if your code works on Windows on a different machine (There are over 300 students, we cannot debug code for you if it doesn't work!)

ASSIGNMENT STEPS

1. **Convert to greyscale and normalize**

Read input image using the '*readRGBImageToSeparatePixelArrays()*' helper function. Convert the RGB image to greyscale (RGB channel ratio 0.3 x red, 0.6 x green, 0.1 x blue). Stretch the values between 0 to 255 (using 5-95 percentile strategy).

Hint 1: see lecture slides 'ImagesAndHistograms' and Coderunner Programming quiz in Week 10.

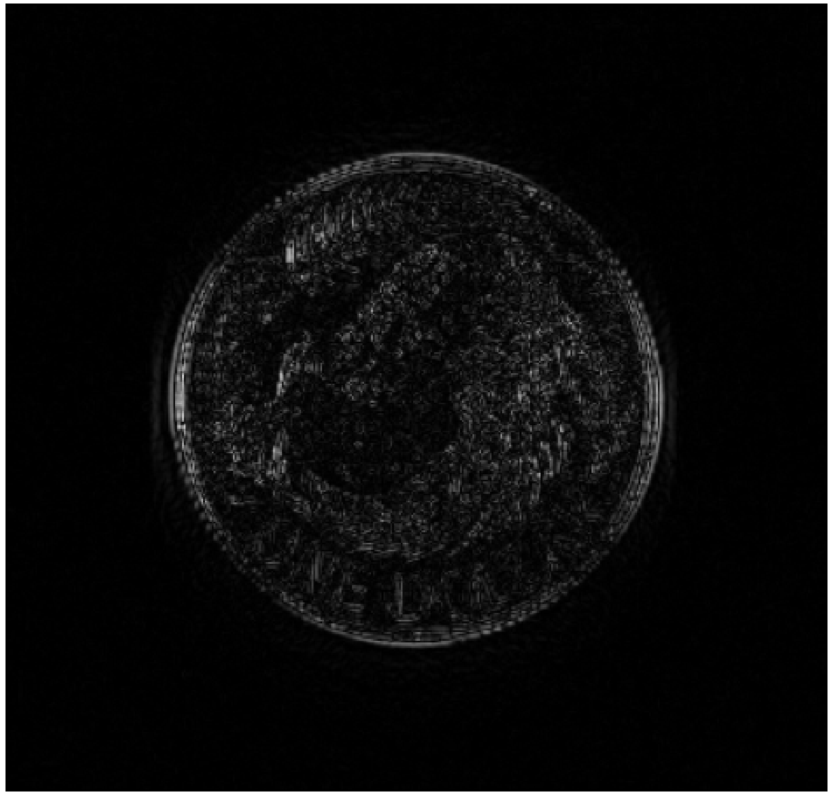


2. Edge Detection

Apply a 3x3 Scharr filter in x- and y-directions independently and take the absolute value of the difference between the results.

Hint 1: see lecture slides on edge detection and Coderunner Programming quiz in Week 11.

Hint 2: you can find the Scharr filter kernel information on this website: https://docs.amd.com/r/en-US/Vitis_Libraries/vision/api-reference.html_2_91

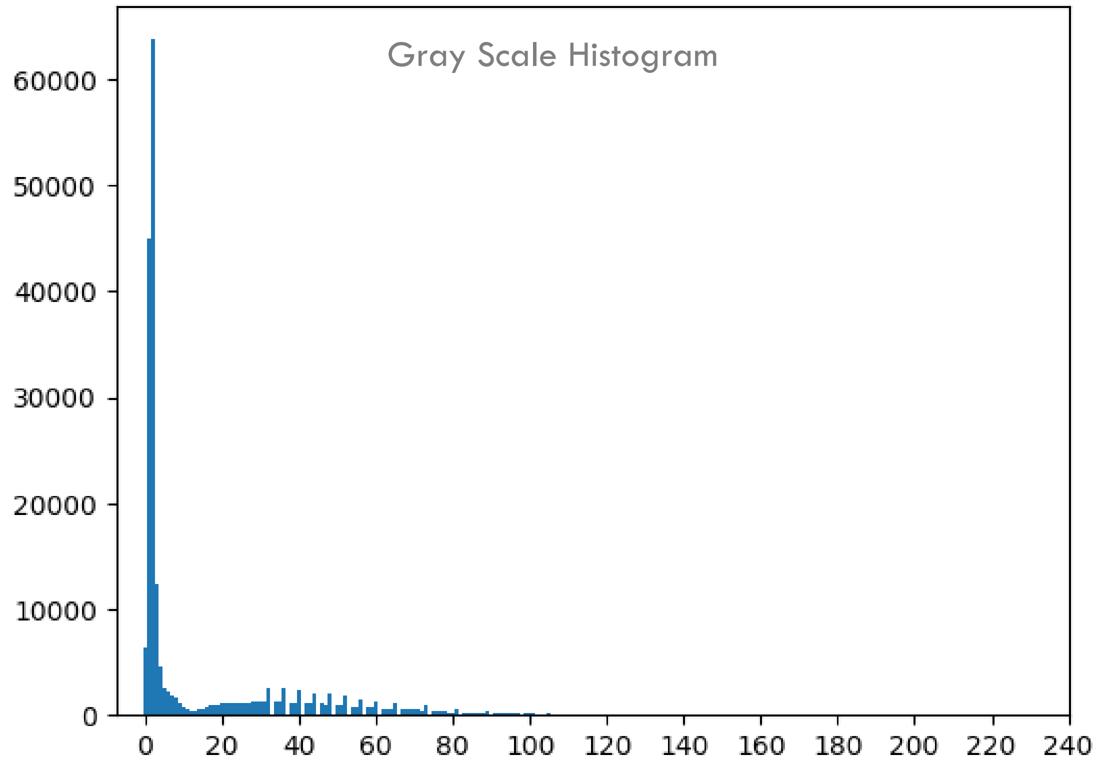


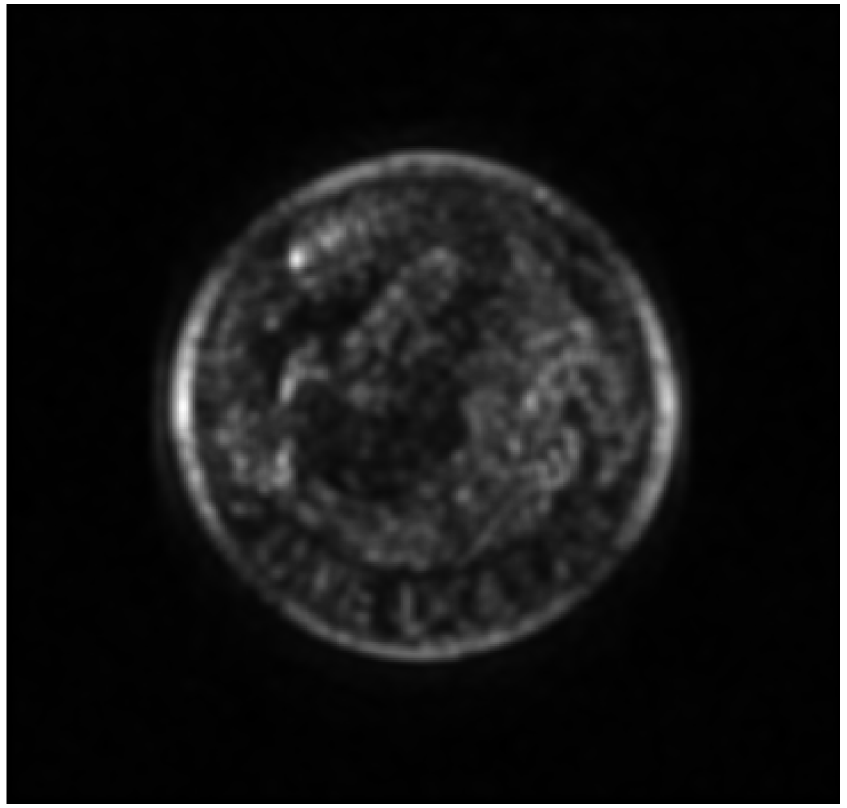
3. Image Blurring

Apply 5x5 mean filter(s) to image.

Hint 1: try applying the filter one or two times to the image sequentially.

Hint 2: see lecture slides on image filtering and Coderunner Programming quiz in Week 11.



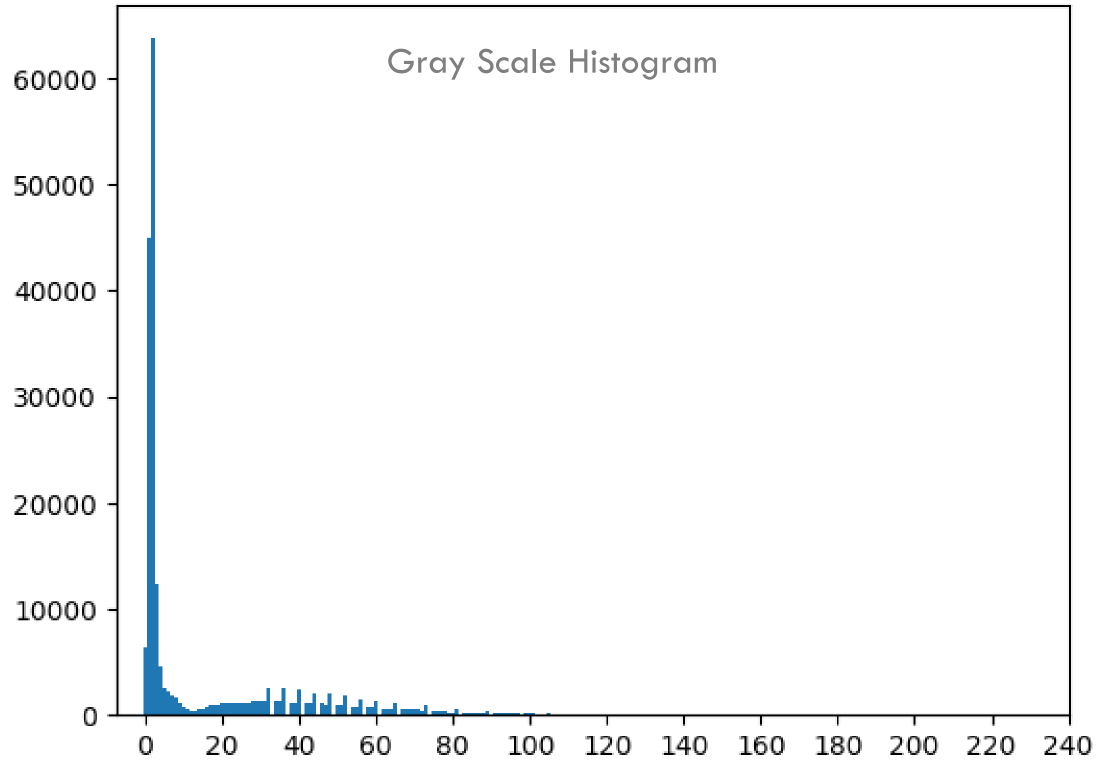


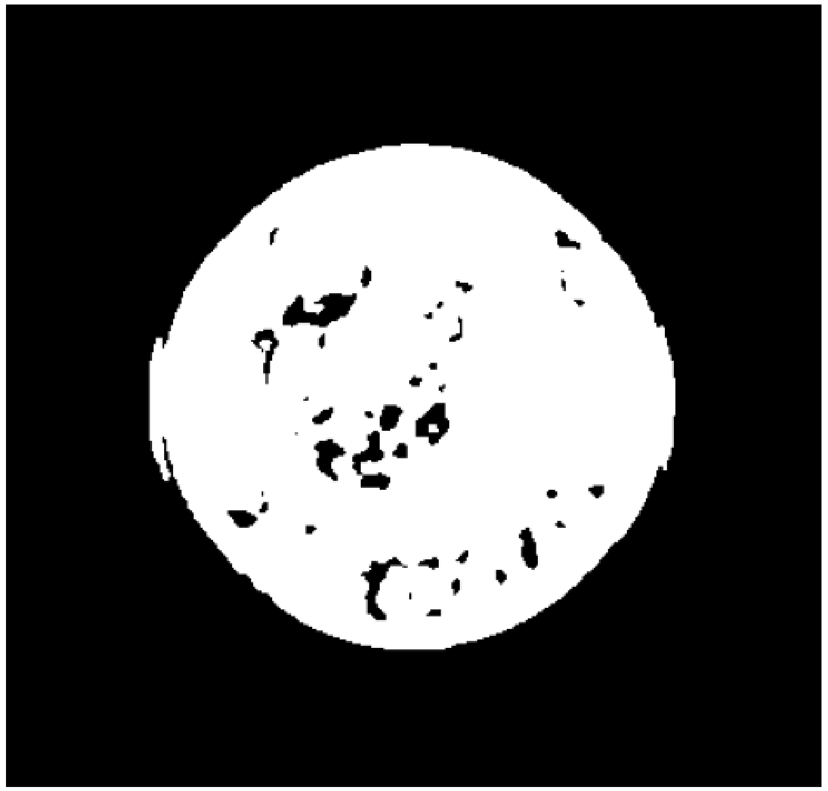
4. Threshold the Image

Perform a thresholding operation to segment the coin(s) from the black background. After performing this step, you should have a binary image.

Hint 1: Two or three would be a reasonable value for thresholding.

Hint 2: See lecture slides on image segmentation and see Programming quiz on Coderunner.





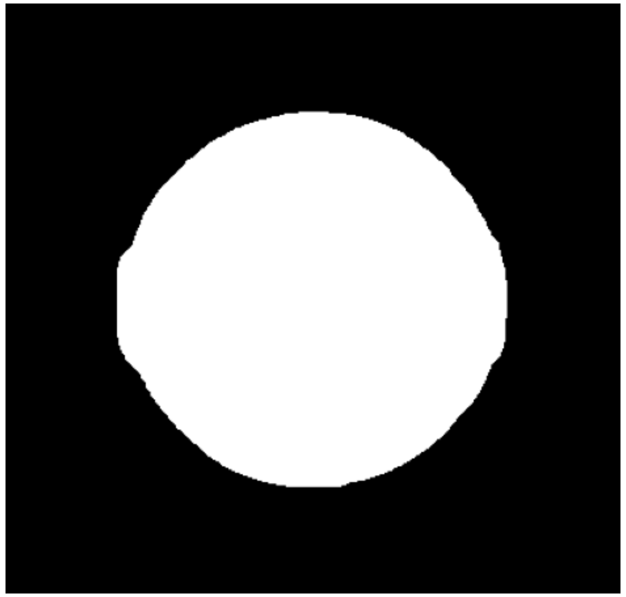
5. Erosion and Dilation

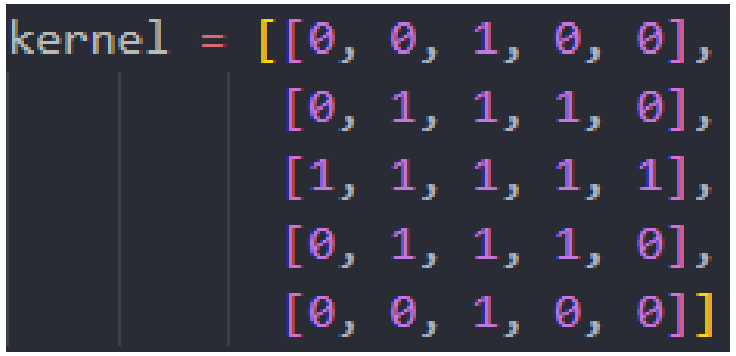
Perform several dilation steps followed by several erosion steps. You may need to repeat the dilation and erosion steps multiple times.

Hint 1: use circular 5x5 kernel, see kernel example on the bottom right image.

Hint 2: try to perform dilation 2-3 times first, and then erosion 3-4 times. You may need to try a couple of times to get the desired output.

Hint 3: see lecture slides on image morphology and Coderunner Programming quiz in Week 12.





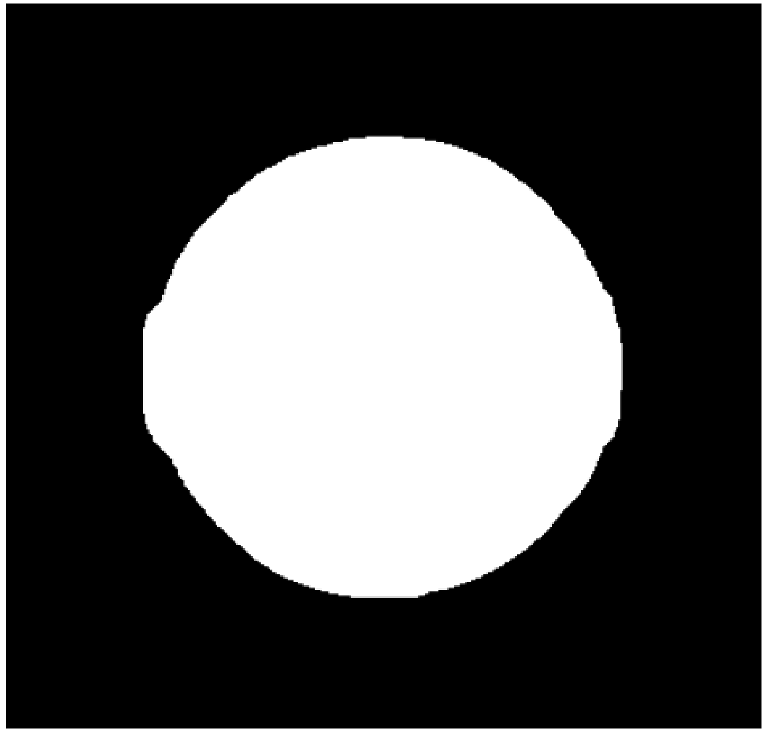
```
kernel = [[0, 0, 1, 0, 0],
          [0, 1, 1, 1, 0],
          [1, 1, 1, 1, 1],
          [0, 1, 1, 1, 0],
          [0, 0, 1, 0, 0]]
```

6. **Connected Component Analysis**

Perform a connected component analysis to find **all** connected components.

After erosion and dilation, you may find there are still some holes in the binary image. That is fine, as long as it is one connected component.

Hint 1: see Coderunner Programming quiz in Week 12.

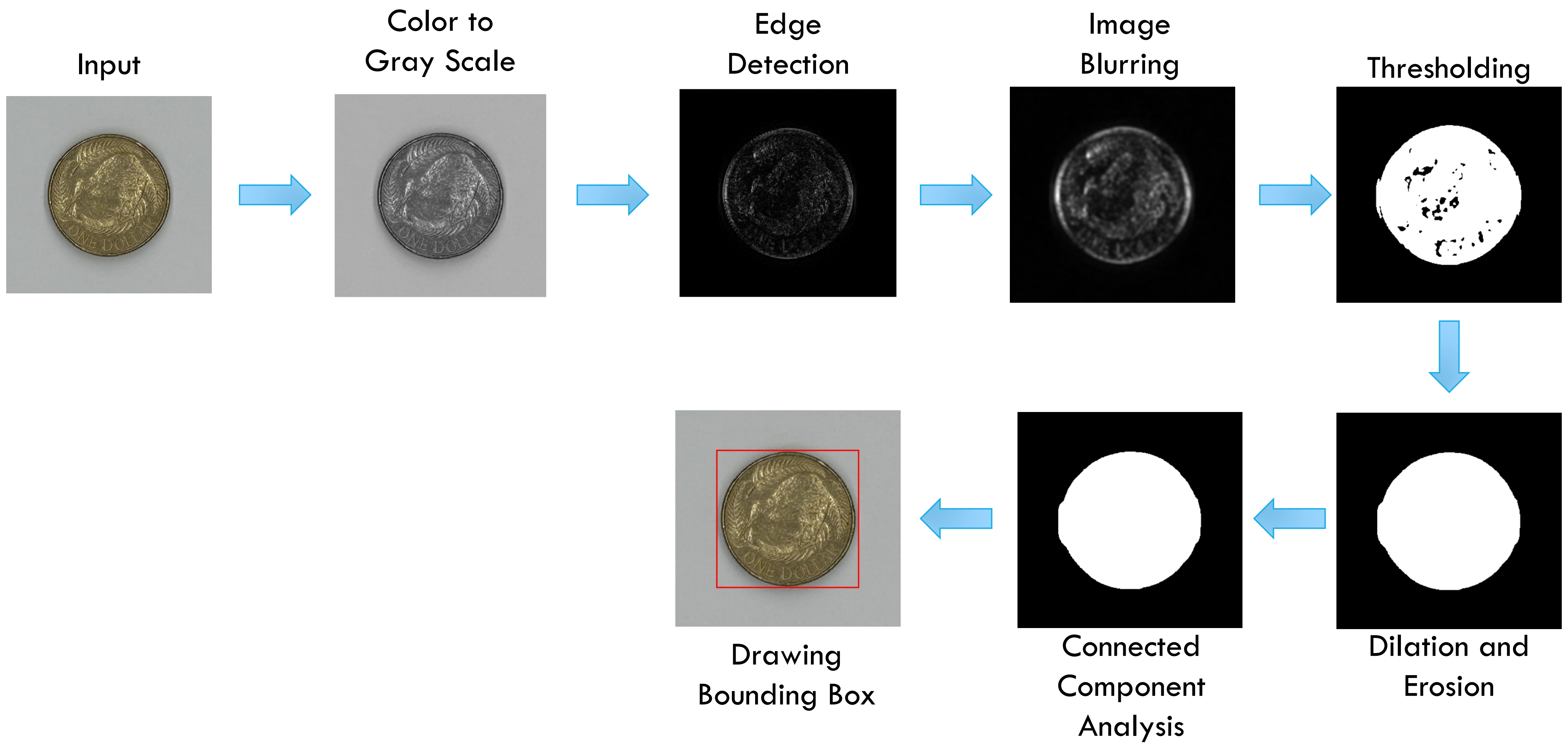


7. Draw Bounding Box

Extract the bounding box(es) around all regions that your pipeline has found by looping over the image and looking for the minimum and maximum x and y coordinates of the pixels in the previously determined connected components.

Make sure you record the bounding box locations for each of the connected components your pipeline has found.

We will provide code for drawing the bounding box(es) in the image, so please store all the bounding box locations in a Python list called ‘*bounding_box_list*’, so our program can loop through all the bounding boxes and draw them on the output image.


Input
Color to Gray Scale
Edge Detection
Image Blurring
Thresholding
Dilation and Erosion
Connected Component Analysis
Drawing Bounding Box

ONE DOLLAR

20

10
TWO DOLLARS
ONE DOLLAR

TWO DOLLARS
20
50
10
ONE DOLLAR

EXTENSION

For this extension (worth 5 marks), you are expected to alter some parts of the pipeline.

- Using Laplacian filter for image edge detection.
- Output number of coins your pipeline has detected.
- Testing your pipeline on the hard-level images we provided.
- Identify the type of coins (whether it is a 1-dollar coin, 50-cent coin, etc.).
  - Since different type of coins have different sizes, you may want to compute the area of the bounding box in the image to identify them.
- Etc.

Submissions that make the most impressive contributions will get full marks. Please create a new Python source file called '*CS373_coin_detection_extension.py*' for your extension part, and include a short PDF report about your extension. Try to be creative!