PRELIMINARY WORK

Objective:

The goal of this lab is to get hands-on experience with basic image and video processing using MATLAB. We were introduced to important concepts like blob detection, morphological operations, edge detection, shape classification, and video frame analysis. I wanted to not only run the scripts but also actually understand what they were doing, how thresholding works, how shapes are identified, and how motion can be tracked across video frames. This lab set the foundation for future work in computer vision and made me comfortable using MATLAB's toolbox features for real world visual tasks.

Environment Setup:

I ran this lab on my MacBook Air with the M2 chip. Installing MATLAB wasn't smooth at first, I ran into server issues while trying to sign up and download it. At one point, the MathWorks website was down, which slowed me down a lot. I also had to install a separate Java runtime because Apple silicon MATLAB requires it. Eventually, I got it all sorted and installed the following toolboxes:

- Image Processing Toolbox
- Computer Vision Toolbox
- Statistics and Machine Learning Toolbox
- Deep Learning Toolbox
- Signal Processing Toolbox

Once everything was set up, I verified the toolboxes using the ver command inside MATLAB and confirmed that I was good to go. This step was important because the lab scripts rely on some of these toolboxes, especially the Image Processing Toolbox.

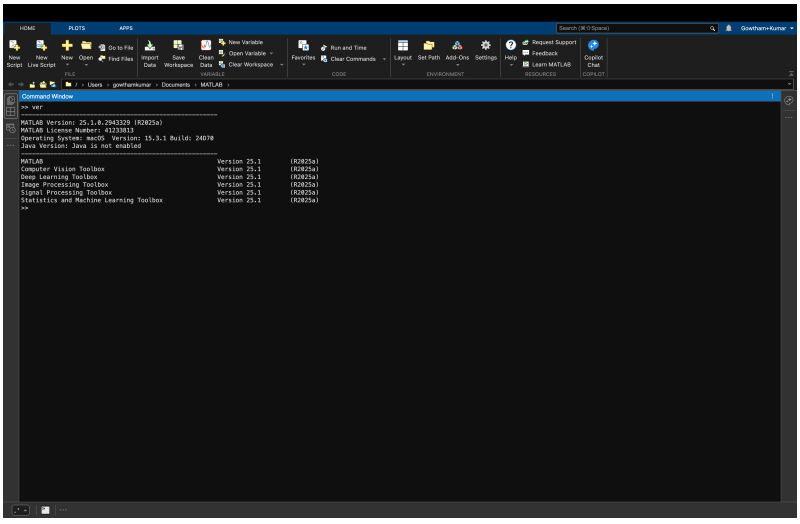

Fig: ver command

Script Preparation:

After getting MATLAB up and running, I created three separate script files for each appendix provided in the lab:

- BlobsDemo.m → Appendix A
- Classify.m → Appendix B
- ExtractMovieFrames.m → Appendix C

I placed all the scripts and input files (like the shape image and video file) inside a dedicated MATLAB folder on my laptop to keep things organized. I also made sure the file names matched what was being referenced in the code, like ensuring the shapes image (.bmp), and the video file (.mp4) were in the same directory or properly linked.

---

1. Review course materials on MATLAB and its image processing toolbox and familiarize yourself with the operations of MATLAB.

I spent some time getting comfortable with MATLAB and how the Image Processing Toolbox works. Honestly, at first it felt overwhelming because of how many functions there are, but once I understood the basics like how to read an image, convert it to grayscale, and apply thresholding or labeling, it started to make sense. I also explored functions like imshow, regionprops, and bwlabel, and I ran them a few times just to see what kind of outputs they give. Getting hands-on helped more than just reading through the docs.

2. Review course materials on the basic concepts of image processing and make sure that you comprehend these concepts and can explain them.

In the lecture recording, we went over image segmentation, edge detection, morphological operations, histogram expansion, object recognition, and video processing. I was familiar with these concepts from class and my previous courses like computer vision, but working with them in MATLAB made things click better. Seeing how the scripts applied these concepts step by step helped me understand how everything connects, from converting images to grayscale, to segmenting them, to actually recognizing patterns or objects. Running the code gave me a more practical understanding of each operation.

3. Check one of the provided scripts and describe what the script does in general.

I went through the script in Appendix A, which is called BlobsDemo.m. It loads the coins.png image and does blob detection using thresholding and connected component labeling. It measures properties like area, intensity, and centroid of each blob (coin), and then filters them based on size and brightness to separate dimes and nickels. The script also visualizes each step — original image, binary mask, color-labeled blobs, and outlines. It helped me understand how regionprops, bwlabel, and label2rgb work together to analyze objects in an image.

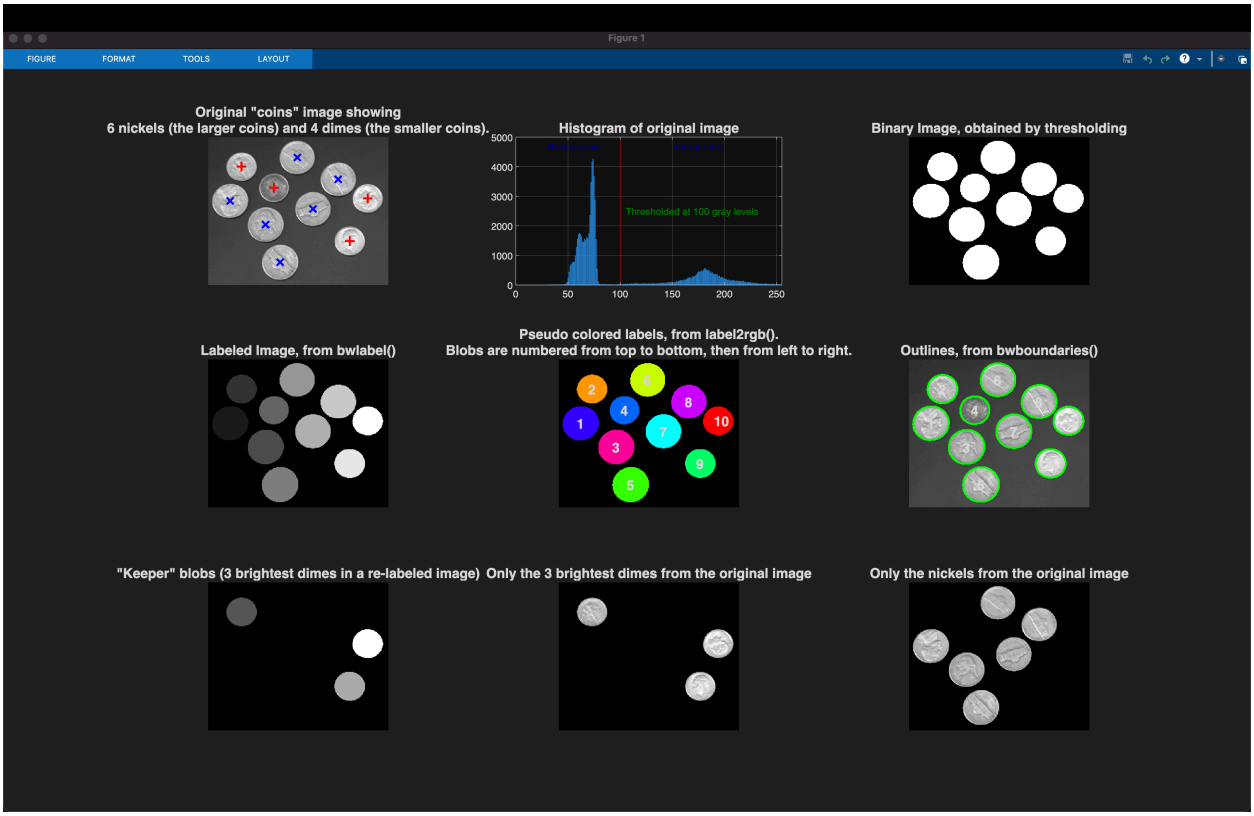

Figure 1: Output of BlobsDemo.m script showing 9-step processing of the coins.png image, from grayscale conversion and thresholding to labeling, boundary detection, and filtering of dimes and nickels based on area and intensity.

4. Prepare several images for image classification, so as to generate more results and have a better understanding when checking them.

I used the shape image provided by the professor (.bmp file) to test the shape classification script, but I also tried modifying the image to include more shapes like ellipses and irregular polygons just to see how the script handled them. It helped me understand how the classification logic works and where it might struggle, for example, it only classifies basic shapes like circles, rectangles, and squares based on aspect ratio and extent. Creating and testing with multiple shapes gave me a better feel for how robust the script actually is.

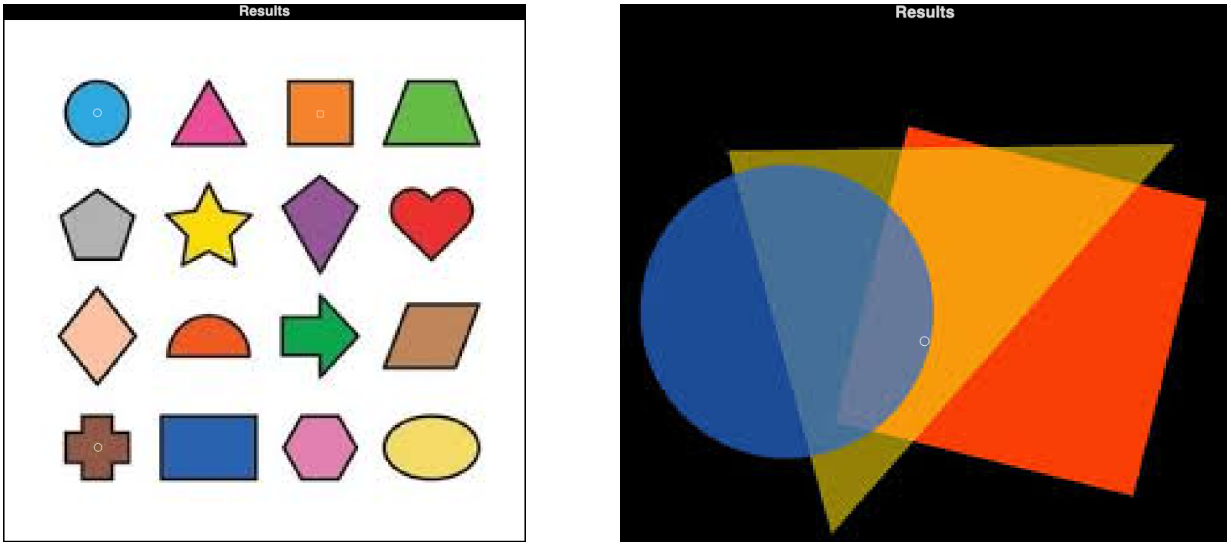

Fig: Test cases

In the first image there are multiple basic and complex geometric shapes (like stars, hearts, pentagons, arrows). The script attempted classification and was able to identify basic shapes like squares and circles based on extent and bounding box ratios. However, more complex or irregular shapes were left unclassified, which shows the limitations of the current shape logic.

The second image with overlapping transparent shapes. Interestingly, the script still managed to classify one region as a circle and placed a white 'O' to mark it. However, I noticed that the white 'O' wasn't placed at the center of the visual blue circle, it was off to the side. After thinking about it, I realized that the script doesn't detect visual shapes directly, it processes connected regions in the binary mask. Because the circle was overlapped by other shapes, the script detected a merged blob, and the centroid it calculated was for that entire irregular region, not the original circle. So technically, it did what it was supposed to, just not how I expected it visually. This actually helped me understand how important image clarity and preprocessing are when doing classification, overlapping or noisy inputs can easily mess with region detection and centroid accuracy.

Student name: **Gowtham Kumar Kamuni**
Student ID: **A20549435**