PRELIMINARY WORK

Objective:

The objective of this preliminary assignment is to establish a solid foundation for Lab 3 by reviewing OpenCV operations and GPU acceleration using CUDA on Jetson Orin Nano. It also involves understanding provided image processing scripts and preparing a clean working environment with appropriate test images and videos. This prep ensures that the actual lab execution, including segmentation, shape recognition, and video annotation, runs smoothly and meets all performance and academic requirements.

Tools and Environment Prepared

- Platform: Jetson Orin Nano running Ubuntu OS
- Language & Libraries: Python 3, OpenCV (cv2), NumPy, Matplotlib
- Test Media Prepared:
  - test.bmp - used for segmentation, morphology, and shape extraction
  - shapes.png - used for shape recognition
  - video.mp4 - main input for video processing in Phase 3
- Project Structure: Maintained a clean and modular folder structure with directories like phase1_segmentation, phase2_shape_extraction, phase2c_morphology_histogram, phase3_output_frames.
- Jetson-Specific Setup: Enabled Python with OpenCV GPU support, verified performance using CUDA enabled routines
- All scripts were pre tested independently before being recorded together during final execution.

1. Review course materials on OpenCV and familiarize yourself with its operations.

I reviewed the basics of OpenCV, especially the core image processing functions that we later used in this lab. I focused on understanding how OpenCV handles grayscale conversion, image thresholding, contour detection, and edge detection methods like Canny and Sobel. These operations are essential for segmenting and analyzing shapes in images, which is a key part of our lab. I also experimented with some OpenCV demo scripts beforehand to get a feel for how different preprocessing steps affect the final output. The hands on review helped me understand how OpenCV translates raw image data into something we can measure, classify, and visualize.

2. Review course materials on the basic concepts of GPU acceleration using CUDA on Jetson Orin Nano, and make sure that you understand why this is beneficial for image processing.

GPU acceleration using CUDA is a big advantage on the Jetson Orin Nano because it offloads heavy computation from the CPU to the GPU, which is built for handling parallel tasks like image processing. I reviewed how CUDA works with OpenCV and learned that certain functions like filtering, thresholding, and morphological operations can be executed much faster on the GPU. This is especially important in real time or video based applications, where frame processing needs to happen quickly.

The Jetson Orin Nano is optimized for this kind of parallel computation, and using CUDA allows us to scale image processing tasks without lag, making our lab scripts run faster and more efficiently.

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

I reviewed the video processing script that extracts frames from a video, analyzes them, and reconstructs a new annotated video. The script starts by reading a sample .avi video and saves each frame as an image. It then calculates the average gray level and RGB values for every frame and plots them to show how brightness and color fluctuate over time.

One of the more interesting parts is the adaptive background modeling, the script creates a dynamic background model that updates slowly over time, and then compares each frame to this background to detect motion. It uses Otsu thresholding to convert this difference into a binary motion mask, highlighting where changes occurred.

Finally, it rebuilds a complete video from the annotated frames. This script goes beyond shape detection, it introduces time based analysis, motion detection, and visualization, making it a powerful tool for surveillance or dynamic scene understanding.

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

For this lab, I prepared and used a variety of images to test different phases of the pipeline.

- For segmentation and thresholding, I used test.bmp to experiment with binary conversions and edge detection.
- For shape classification, I used shapes.png which includes clearly defined geometric figures like triangles, circles, and rectangles, matching the logic in Appendix A.
- For morphological operations, the same test.bmp was reused to observe how opening, closing, dilation, and erosion transform the shape boundaries.
- For video processing, I used video.mp4 which was a short test video with visible moving shapes. It allowed me to extract frames and apply real-time shape labeling on video.

Preparing these test inputs helped validate the scripts under different scenarios and made debugging easier during execution.

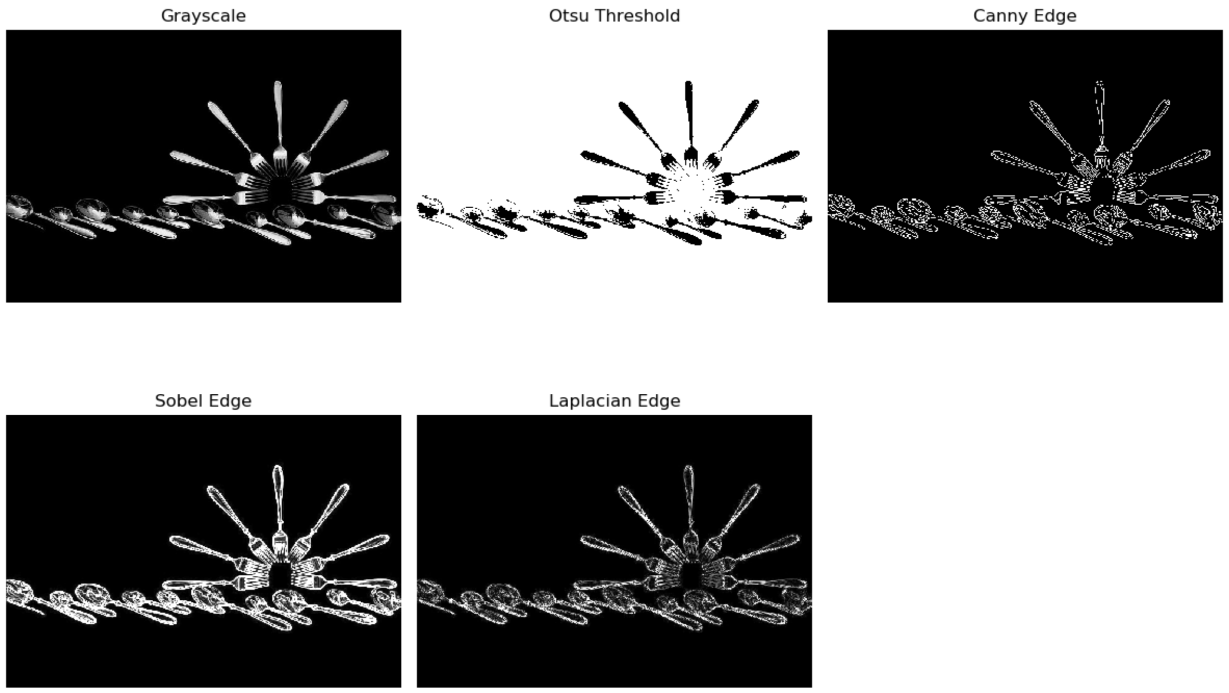

Fig: Image Processing Stages – Segmentation and Edge Detection

This figure demonstrates the progression of image processing applied to a grayscale image of spoons and forks using different OpenCV techniques:

- Grayscale: The original image is converted to grayscale, reducing complexity by removing color channels while preserving structural detail.
- Otsu Threshold: A global thresholding method is applied to segment the image into binary format (black and white), effectively isolating foreground objects.
- Canny Edge: Highlights precise edges by detecting gradients and suppressing noise, producing a clean edge map.
- Sobel Edge: Captures horizontal and vertical gradients, emphasizing sharper boundaries and object outlines.
- Laplacian Edge: Detects rapid intensity changes and enhances fine details in regions with curvature and corners.

These variations help build a deeper understanding of how different edge detection and segmentation methods affect object visibility and clarity, crucial for preprocessing in shape detection and classification tasks.

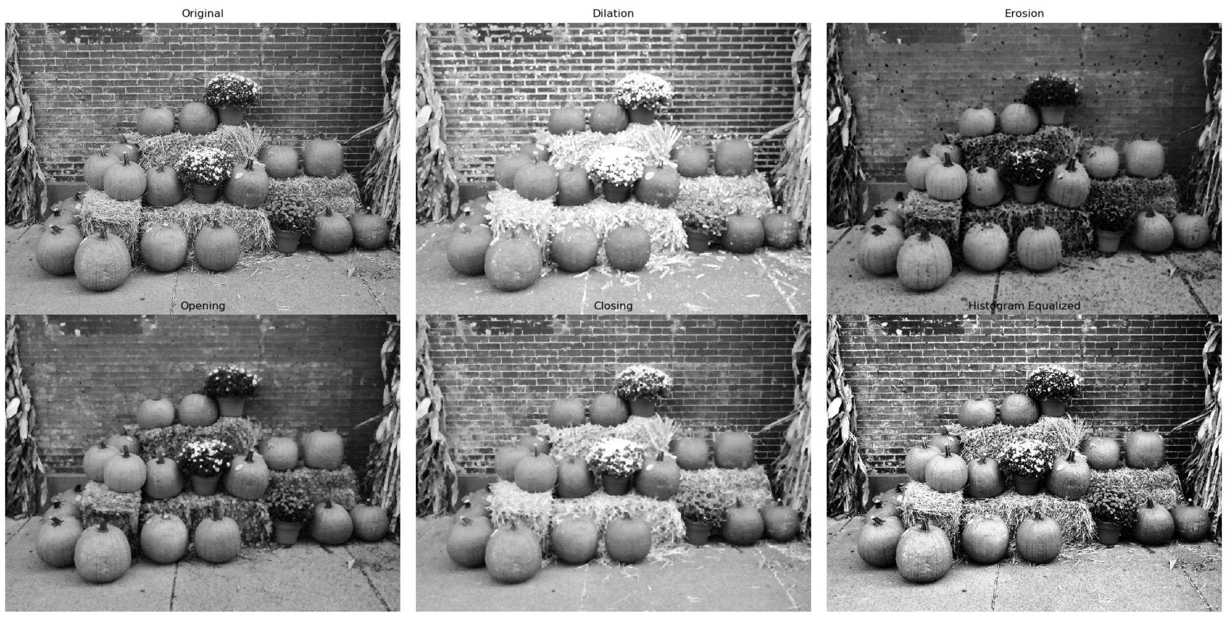

Fig: Morphological Operations and Histogram Equalization

This figure demonstrates a series of morphological transformations and contrast enhancement techniques applied to a grayscale image of pumpkins and hay:

- Original: The untouched grayscale image used as the base reference.
- Dilation: Expands the white regions (foreground), useful for connecting broken parts of an object.
- Erosion: Shrinks the white regions, eliminating small white noise and detaching connected objects.
- Opening: Erosion followed by dilation, effective in removing small objects or noise while preserving the main shape.
- Closing: Dilation followed by erosion, fills small holes and gaps inside the foreground objects.
- Histogram Equalized: Enhances image contrast by redistributing pixel intensity values across the histogram, making features stand out more clearly.

These transformations are key for preprocessing in image classification tasks, helping the system distinguish shapes, edges, and texture under varying conditions.

This preliminary work provided an essential foundation for understanding core image processing operations using OpenCV and the CUDA-accelerated Jetson Orin Nano platform. By reviewing key concepts, exploring provided scripts, and generating experimental results, I gained practical insights into morphological transformations, edge detection, and frame-based analysis. The included images illustrate both edge-based and morphology-based techniques applied to sample inputs. The rest of the lab execution, including implementation details, scripts, and final outputs, are presented comprehensively in the accompanying lab report.

Student name: Gowtham Kumar Kamuni
Student ID: A20549435