TAG
DYNAMICS

Real-time OpenCV Video & Image Processing

Author: *Haitham Sedqi*

This project focuses on real-time video and image processing using **OpenCV** (Open Computer Vision) library in **Python 3.12**.

OpenCV provides a large arsenal of optimized algorithms for image processing and analysis, all accessible through a Python interface. Real-time processing of data is critical for efficiency in applications such as autonomous vehicles, industrial manufacturing systems, medical imaging systems, among others.

In this project, computer vision techniques such as **grayscale conversion, Gaussian blur, and Canny Edge Detection** are leveraged for both static images and live video streams. By using the OpenCV library, we utilize optimized algorithms and real-time capabilities to implement image and video processing tasks, as required in this project.

Tools used: **Python 3.12, OpenCV, VS Code**

Task #1: Image Processing

The goal of task #1 is to process a static image using **grayscale conversion**, **Gaussian blur**, and **Canny edge detection** to extract the boundaries and structures in that respective image. Also, we should output every image through each phase (Original image, grayscale image, blurred image, and edge detection image) simultaneously and on different windows.

To perform task #1, we completed the following steps:

1. Convert colored image to grayscale image

2. Apply Gaussian blur on the grayscale image to get the blurred image

3. Apply Canny Edge Detection on the blurred image to get the edge detected image as the final result.

Challenges:

Initially, I used an image of a **2015 S-Class Mercedes** for edge detection, and it yielded nearly identical results across different thresholds, this was due to less variations in the edges of the shape. To address this, I used instead an image of a **1950 Mercedes SL Roadster**, with more shape and intensity variations, and it resolved this issue. The algorithm operated as expected, and I observed how an object's structure can naturally affect edge detection.

Task #1: Results

Important parameters:

Kernel Size = (5,5)

Lower Threshold = 50

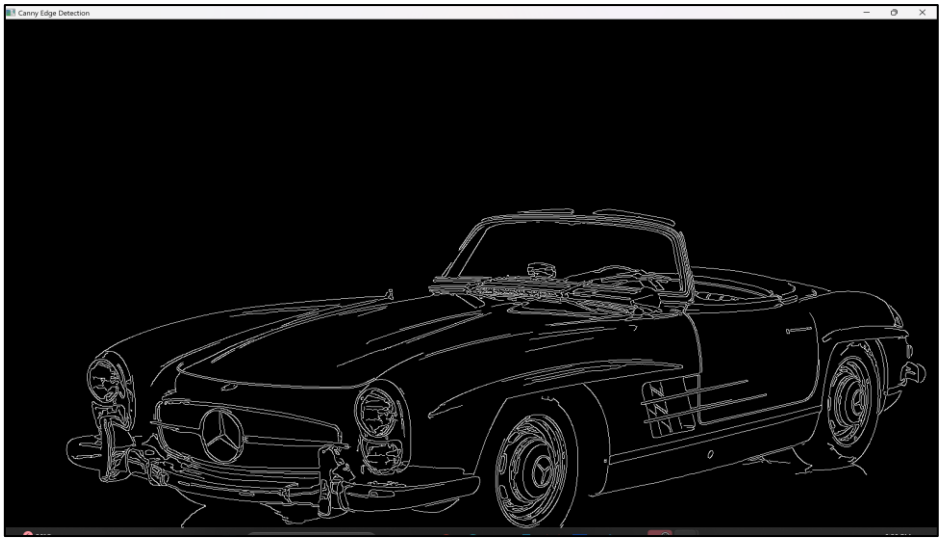
Upper Threshold = 150

Figure 1: Original Image

Figure 2: Grayscale image

Figure 3: Blurred Image

Figure 4: Canny Edge Detection



Task #2: Real-time Video Processing

The goal of task #2 is to process a live video stream (by webcam) by using **grayscale conversion**, **Gaussian blur**, and **Canny edge detection** to extract edges & shapes in real-time. Similar to task #1, each processed phase of a video (original, grayscale, blurred, edge-detected) must be presented simultaneously on a separate pop-up window.

To perform task #2, we completed the following steps:

1. Capture video from webcam

2. Convert each frame in the stream to grayscale

3. Apply Gaussian Blur

4. Apply Canny Edge Detection

5. Display all processed video phases simultaneously

Challenges:

When experimenting with high threshold values, e.g. 500 and above, I expected not to find any edges. Ironically, I found the pop-window to contain edges, and it was strange to me at the beginning. After some thorough research, I understood that edge calculation is based on gradient intensity, not raw pixel values (limited to 0->255). If a sharp change in intensity across neighboring pixels occurs, the G will yield a very high number & an edge will appear, regardless of the thresholds.

Task #2: Results

Kernel size = (3,3)

Lower threshold = 50

Upper threshold = 150

Figure 5: Original webcam

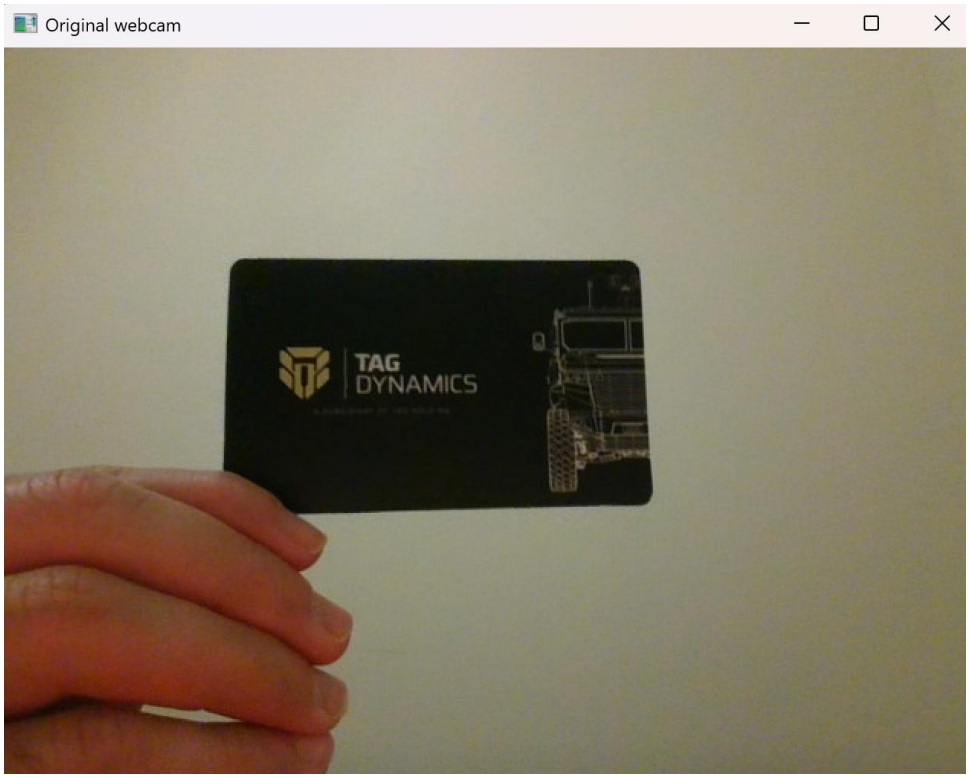


Figure 6: Grayscale webcam

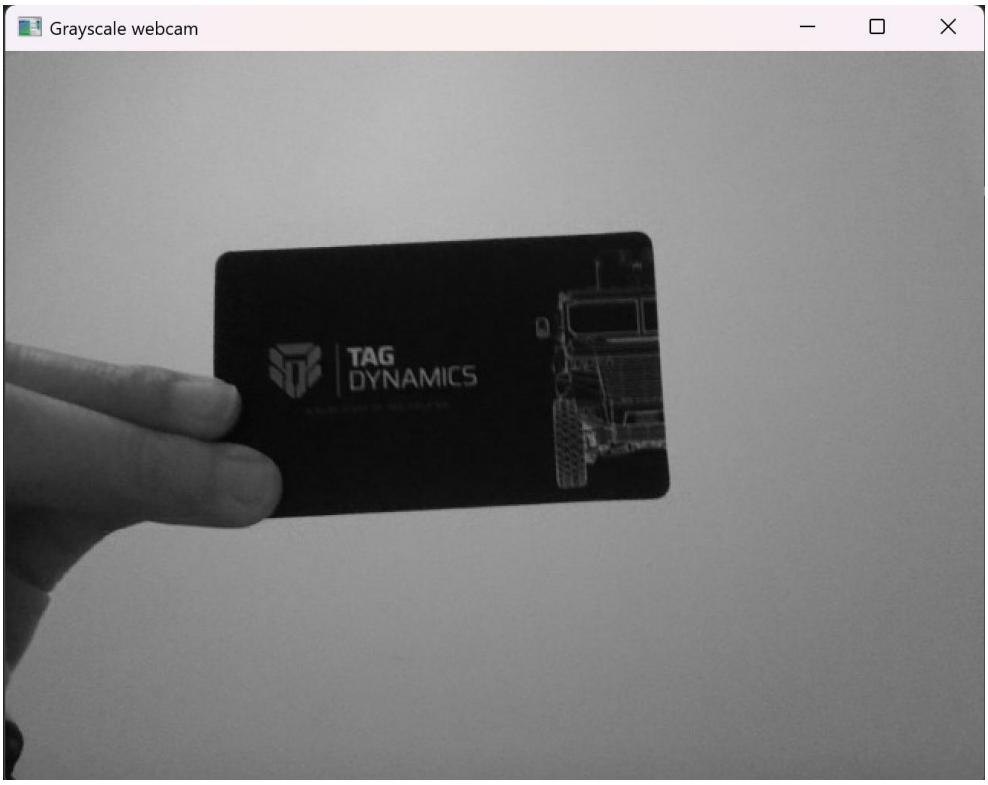


Figure 7: Blurred webcam

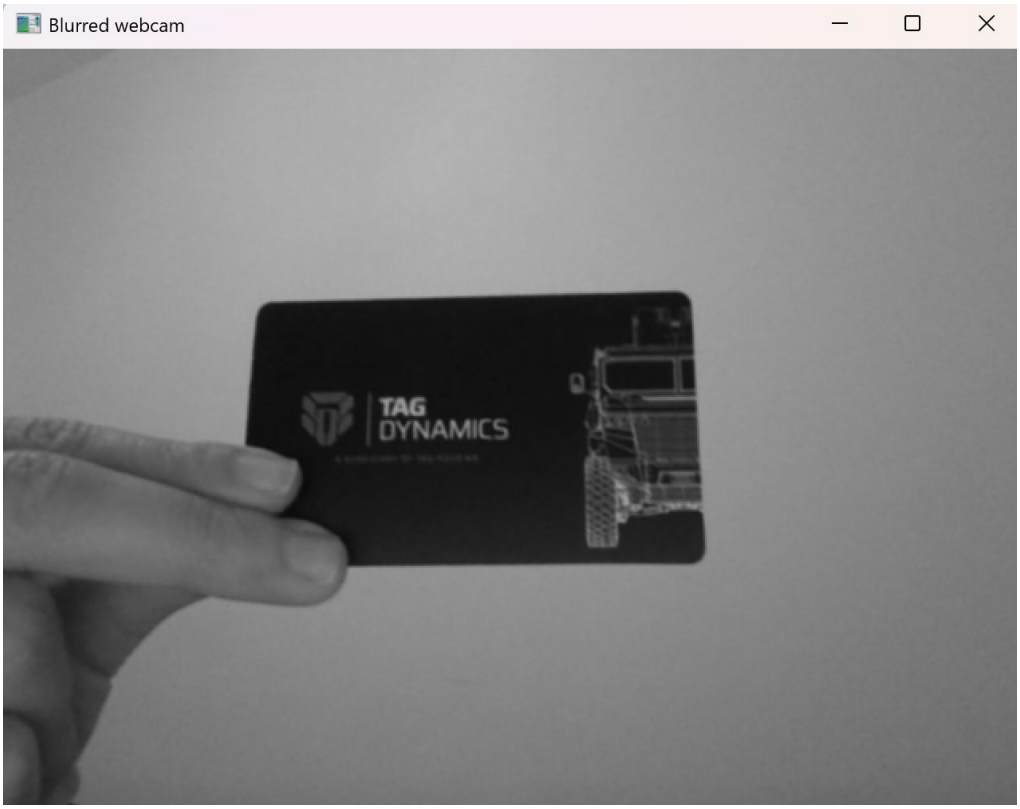
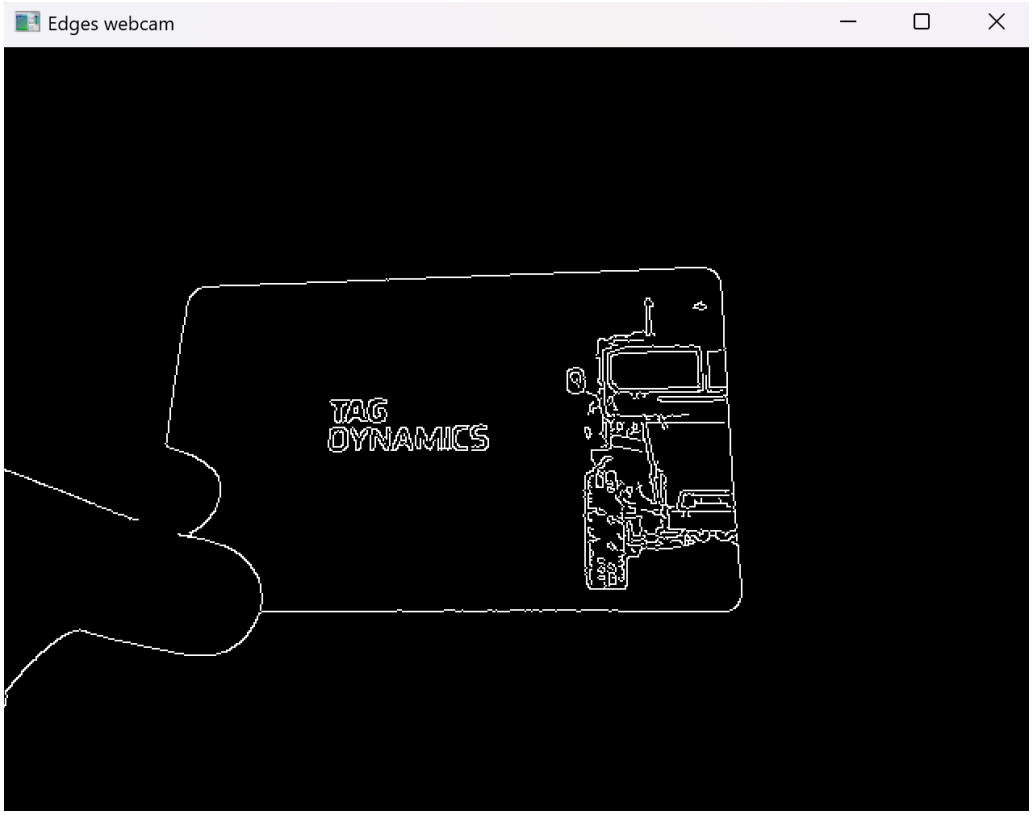


Figure 8: Edges webcam



This project demonstrated real-time image and video processing using OpenCV in Python 3.12. By implementing a concatenation of computer vision algorithms, we extracted edges from both static images and live video streams. The structured approach allowed for visualization of each processing stage, showcasing how edge detection can be used to highlight boundaries and structures.