Giorgio Mendoza
CS549-E24-E01
Dr. Alexandros Lioulemes

Week 5 Lab

Part 1:

Part 1 of this project, VisionCoin, aimed to develop an application that could interact with US coins in three ways. The first goal was to detect coins from a camera's view, focusing on identifying circular shapes that could represent coins on a surface. Following detection, the application needed to recognize different types of coins such as quarters, dimes, nickels, and pennies by analyzing specific features from the camera input. The final task involved calculating the total value of all recognized coins and displaying this amount on the screen, demonstrating the application's ability to combine image recognition with real-time data processing.

Code Explanation:

First, we start by loading the coin reference images in grayscale, which are important for matching and recognition tasks.

For each coin type (quarter, nickel, penny, dime), there are dedicated functions (e.g., detect_quarter_circles) that apply image processing techniques like grayscale conversion and median blurring to enhance image quality for better circle detection using the Hough Circle Transform. The dedicated functions were needed because each coin has different parameters. This method detects the circular shapes of coins from the video frames.

The SIFT algorithm was selected for this project; functions like find_keypoints_and_descriptors extract keypoints and their descriptors from images. These are necessary for recognizing coins by comparing descriptors of live capture with those of reference images through match_descriptors, which uses a FLANN-based matcher.

The process_frame function integrates all these operations to process live video frames. It detects coins, identifies them by comparing with reference images, and annotates the frames with detection results and coin types. It also keeps track of the total value of detected coins to display it on the screen, ensuring each coin type is counted once to avoid duplications in value calculations.

The main function sets up the video capture, iterates through frames, and processes each frame using the aforementioned logic. It also handles the real-time display of results and terminates the program when requested.

Results:

The code's ability to recognize coins depended heavily on careful adjustment of its settings, which required a lot of testing and refinement. To aid this process, it was important to create a simple setup with steady lighting and a stable camera to capture the reference images. This setup was crucial for ensuring the reference images were clear and matched well with the live video feed, allowing for accurate parameter tuning.

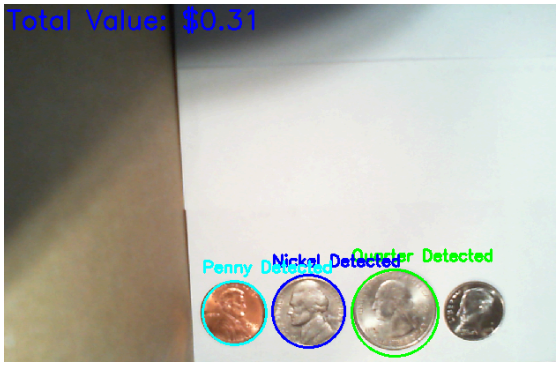

Figure 1. Input Images

Part 2:

The aim is to generate a high dynamic range (HDR) image by merging the given photographs taken at different exposures. This technique captures a broader luminance range than possible with a single exposure.

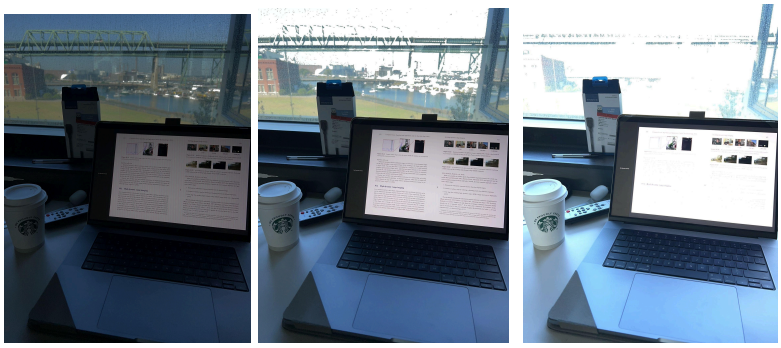

Figure 2. Input Images

Code Explanation:

The code starts by setting up file paths for three images with different exposures and loads them into memory using OpenCV's cv.imread as usual.

Using OpenCV's Mertens Fusion algorithm, the code merges these images into a single HDR image that captures a wider range of luminance. This HDR image, initially in floating-point format, is then converted to an 8-bit format to make it compatible with standard display devices.

The resulting 8-bit image is then saved to a specified path on the computer and we can compare it to the desired output image.

Results:

This part was very straightforward and the resulting output image closely matches the desired reference image, likely due to the high optimization of OpenCV's APIs for tasks like this.

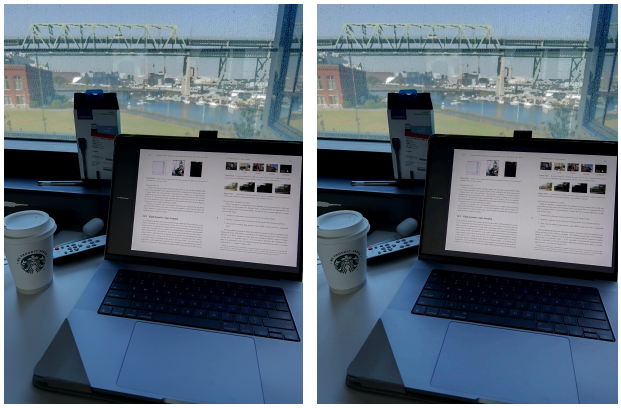

Figure 3. Result Image (left) & Desired Image (right)