**Assignment 4**

CP467: Image Processing & Recognition

Professor: Dr. Zia Ud Din

Due Date: Tuesday November 14, 2023. 11:59 p.m.

Chandler Mayberry 190688910

Table of Contents

[Introduction 3](#_Toc150760458)

[Task 01 3](#_Toc150760459)

[Task 02 5](#_Toc150760460)

[Task 03 7](#_Toc150760461)

[Works Cited 10](#_Toc150760462)

# Introduction

All the code for this assignment may be run using the a04\_190688910.py file in the source code folder. Running this file will perform the operations and display the resulting images to the user using python OpenCV.imshow() method.

# Task 01

For Task 01, I chose three images instead of two to perform key point detection as I wanted to see the performance on another image. The feature detection technique I decided to use was Harris corner detection to find the key point corners in my selected images. The implementation used OpenCV built-in cv2.cornerHarris to perform the Harris operation. The results, although not perfect, do find the major corners in each of the images with the exception of low-contrast areas which fail to find the corners (such as the top left corner of the mirror) and noisy areas (many corners in Figure 2). Instead of manually annotating the image I dilated the corners, so they are easier to find. See the implementation and the results below:

A grey building with windows

Description automatically generated

Figure 1: Building Key Point Detection 1

A tall building with windows

Description automatically generated

Figure 2: Building Key Point Detection 2

A picture frame and a plant in a pot

Description automatically generated

Figure 3: Table Key Point Detection 3

# A screenshot of a computer program Description automatically generated

Figure 4: Feature Detection Implementation

# Task 02

For Task 02 I chose an image of a table and a plant to place on top of the table. After looking at the provided resources I chose to go with the seamless photo blending shown in the third resource [here](https://cs.brown.edu/courses/csci1950-g/results/proj2/edwallac/). To implement this seamless blending method (using the Poisson equation stated in the article) I decided to go with the OpenCV built-in cv2.seamlessClone. This built-in performs the Poisson equation and provides three different flags to use in blending images: normal\_clone, mixed\_clone, and monochrome\_transfer. I found the best result was with the normal\_clone. Mixed\_clone produced an image with the plant being transparent and monochrome\_transfer produced a monochrome plant instead of keeping the regular RGB color space.

Unlike tasks 1 and 3 I found this task to take the most time and was the trickiest as no matter what I seemed to do the plant would be transparent or non-visible in all cases. I made several different attempts at fixing this issue while looking at and contemplating using a pyramid approach toward the end. In the documentation, it states that the mask should be manually selected so I tried to make a rough polygon around the plant, but this did not fix my issue and resulted in a worse image. I then tried making a mask that is only the area around the plant by using cv2.threshold and dilating the edges produced similar to getting the final alternative result in my Marr-Hildreth implementation in assignment 2. This produced a slightly better result but did not fix the transparency issue. After several other attempts to adjust values, contrast, brightness, etc., of the images I was unable to produce a better result.

My problem all along after looking back at the given resource article was simply that my mask was inverted. Where I should have had white values for the foreground, I had black values and vice-versa. Once I made this change in code my result worked as expected and produced a relatively ideal result aside from the blurring around the edges of the plant caused by the dilated mask. I did not perform filtering on the image to sharpen these blurred edges as I feared something like a Laplacian filter may ruin other aspects of the image if applied (such as the noise from small details in the leaves). See the code and results below:

A potted plant on a table

Description automatically generated

Figure 5: Simple Image Manipulation

A screenshot of a computer program

Description automatically generated

Figure 6: Simple Image Manipulation Implementation

# Task 03

For Task 03 the implementation was straight forward using the OpenCV built-in cv2.Stitcher. After creating a stitcher object all that needed to be done was to call the stitch method with the 2 images as input parameters (omitting any masks as they seemly are not required) and it returns a panoramic image stitched together. See the code and results below:

A screenshot of a computer

Description automatically generated

Figure 7: Both Halves of Panorama

A large rock formation in a desert

Description automatically generated

Figure 8: Stitched Image

A computer screen shot of a program

Description automatically generated

Figure 9: Image Stitching Implementation

# Works Cited

[modern apartment building business facade windows commercial place of work on blue sky 21491138 Stock Photo at Vecteezy](https://www.vecteezy.com/photo/21491138-modern-apartment-building-business-facade-windows-commercial-place-of-work-on-blue-sky)

[close up of modern buildings 7567208 Stock Photo at Vecteezy](https://www.vecteezy.com/photo/7567208-close-up-of-modern-buildings)

[Standing picture frame with small potted plants 3384176 Stock Photo at Vecteezy](https://www.vecteezy.com/photo/3384176-standing-picture-frame-with-small-potted-plants)

[Wooden desk for work space 10098080 Stock Photo at Vecteezy](https://www.vecteezy.com/photo/10098080-wooden-desk-for-work-space)

[Home plant in flower pot isolated. Illustration AI Generative 24399214 Stock Photo at Vecteezy](https://www.vecteezy.com/photo/24399214-home-plant-in-flower-pot-isolated-illustration-ai-generative)

[Beautiful Cappadocia on the background of blue sky with white 6570313 Stock Photo at Vecteezy](https://www.vecteezy.com/photo/6570313-beautiful-cappadocia-on-the-background-of-blue-sky-with-white)