UNIVERSITY OF ARKANSAS

CSCE 57003: Computer Vision

Homework #3

Submission Deadline: 11:59 PM, November 20th, 2024

Student ID: 010936980

Student Name: Sankalp Pandey

Late Days Used: 2

Submission Instruction

Read all the instructions below carefully before you start working on the homework, and before submission.

- Your submission contains 2 parts: PDF file and source code folder. All is zipped in a single file. The zip file should be named with your student ID.
- If you have questions, please contact Khoa Vo: khoavoho@uark.edu

Question 1: Image Stitching (Total 50 points)

Overview

In this question, feature-based homography transformation is used to stitch pair or multiple images. This report consists of two parts. Part 1 addresses stitching between a single pair of images and Part 2 addresses stitching among multiple images. Algorithms and parameters are discussed in each section. Two extracredits problems are analyzed in the end.

Question 1.1 (25 points): Stitching A Pair of Image

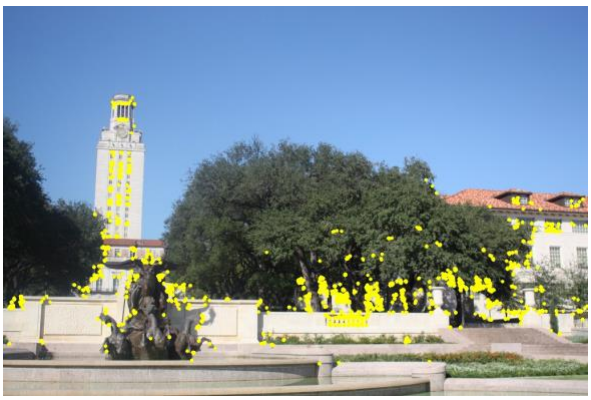
A test set consisting of two images (uttower left.jpg and uttower right.jpg) is to be stitched. The procedures are reviewed in detail below.

```
def stitch_pair(img_src_path: str, img_des_path: str) -> np.ndarray:
    # Load images and convert to grayscale and np.float32
    A = cv.imread(img_src_path)
    B = cv.imread(img_des_path)
    A_gray = cv.cvtColor(A, cv.COLOR_BGR2GRAY).astype(np.float32)
    B_gray = cv.cvtColor(B, cv.COLOR_BGR2GRAY).astype(np.float32)

    # Feature points using Harris corner detector
    threshold = 0.05
    radius = 2
    A_harris = cv.cornerHarris(A_gray, blockSize=2, ksize=3, k=0.04)
    B_harris = cv.cornerHarris(B_gray, blockSize=2, ksize=3, k=0.04)

    # Threshold
    A_corners = np.argwhere(A_harris > threshold * A_harris.max())
    B_corners = np.argwhere(B_harris > threshold * B_harris.max())

    # Corners to KeyPoint objects
    A_keypoints = [cv.KeyPoint(float(c[1]), float(c[0]), radius*2) for c in A_corners]
    B_keypoints = [cv.KeyPoint(float(c[1]), float(c[0]), radius*2) for c in B_corners]
    A_features = draw_features(A, A_keypoints)
    B_features = draw_features(B, B_keypoints)
    cv.imwrite('/Users/sanpandey/Computer Vision/Assignment3/stitchPair/features_A.jpg', A_features)
    cv.imwrite('/Users/sanpandey/Computer Vision/Assignment3/stitchPair/features_B.jpg', B_features)

    # SIFT
    sift = cv.SIFT_create()
    A_kp, A_des = sift.compute(A_gray.astype(np.uint8), A_keypoints)
    B_kp, B_des = sift.compute(B_gray.astype(np.uint8), B_keypoints)
    bf = cv.BFMatcher(cv.NORM_L2, crossCheck=False)
    matches = bf.match(A_des, B_des)

    # Putative matches
    matches = sorted(matches, key=lambda x: x.distance)
    putative_matches = matches[:200]

    # Draw putative matches in blue
    putative_viz = draw_matches(A, A_kp, B, B_kp, putative_matches, color=(255, 0, 0))  # Blue
```

```
cv.imwrite('/Users/sanpandey/Computer Vision/Assignment3/stitchPair/putative_matches.jpg', putative_viz)

# RANSAC
src_pts = np.float32([A_kp[m.queryIdx].pt for m in putative_matches]).reshape(-1, 1, 2)
dst_pts = np.float32([B_kp[m.trainIdx].pt for m in putative_matches]).reshape(-1, 1, 2)

H, mask = cv.findHomography(src_pts, dst_pts, cv.RANSAC, 0.3, maxIters=5000)

# Draw inlier matches in green
inlier_viz = draw_matches(A, A_kp, B, B_kp, putative_matches, mask.ravel().tolist(), (0, 255, 0))  # Green
cv.imwrite('/Users/sanpandey/Computer Vision/Assignment3/stitchPair/inlier_matches.jpg', inlier_viz)

# Panorama
height_A, width_A = A.shape[:2]
height_B, width_B = B.shape[:2]

corners_A = np.float32([[0, 0], [0, height_A], [width_A, height_A], [width_A, 0]]).reshape(-1, 1, 2)
corners_B = np.float32([[0, 0], [0, height_B], [width_B, height_B], [width_B, 0]]).reshape(-1, 1, 2)

transformed_corners_A = cv.perspectiveTransform(corners_A, H)
all_corners = np.concatenate((transformed_corners_A, corners_B), axis=0)

[x_min, y_min] = np.int32(all_corners.min(axis=0).ravel() - 0.5)
[x_max, y_max] = np.int32(all_corners.max(axis=0).ravel() + 0.5)
translation_dist = [-x_min, -y_min]

H_translation = np.array([[1, 0, translation_dist[0]],[0, 1, translation_dist[1]],[0, 0, 1]])
panorama_size = (x_max - x_min, y_max - y_min)
panorama = cv.warpPerspective(A, H_translation.dot(H), panorama_size)
panorama[translation_dist[1]:translation_dist[1]+height_B, translation_dist[0]:translation_dist[0]+width_B] = B
return panorama
```

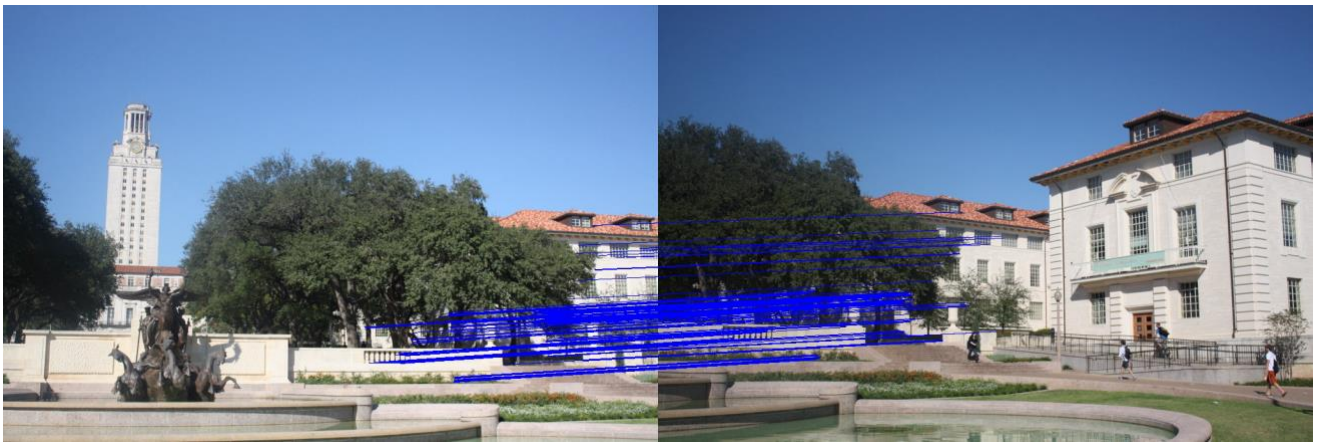
Results:

Features A and B respectively:



Inlier matches:

Putative matches:

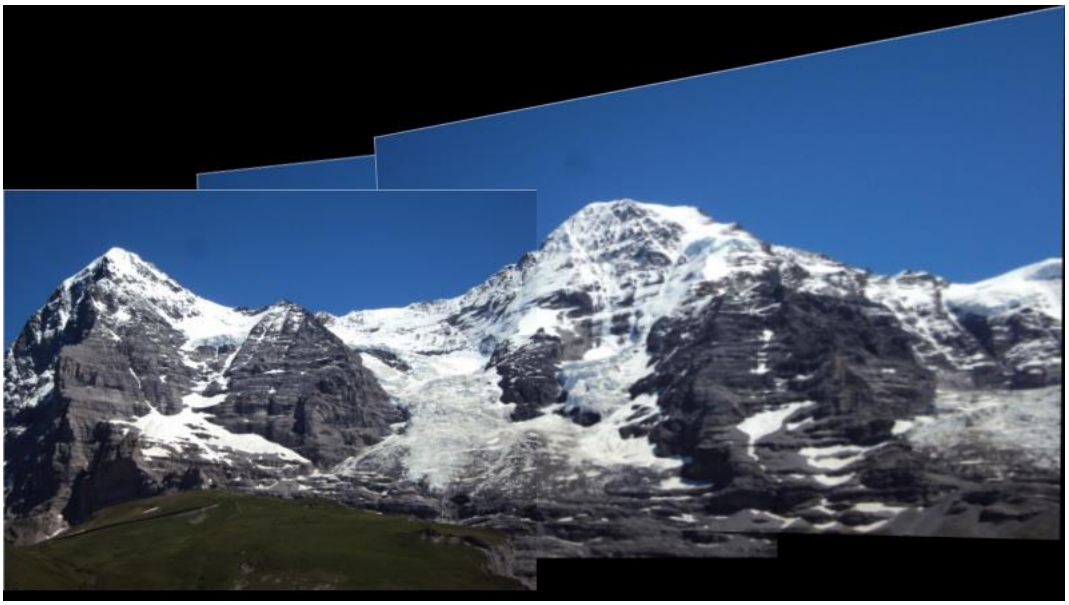


Panorama:

Question 1.2 (25 points): Stitching Multiple Images

Implemented functions as per the described tasks are in the attached source code folder. The results follow. Some of the others may have issues due to Harris Corner detection not being robust enough to match the images properly.

Result for the hill images:



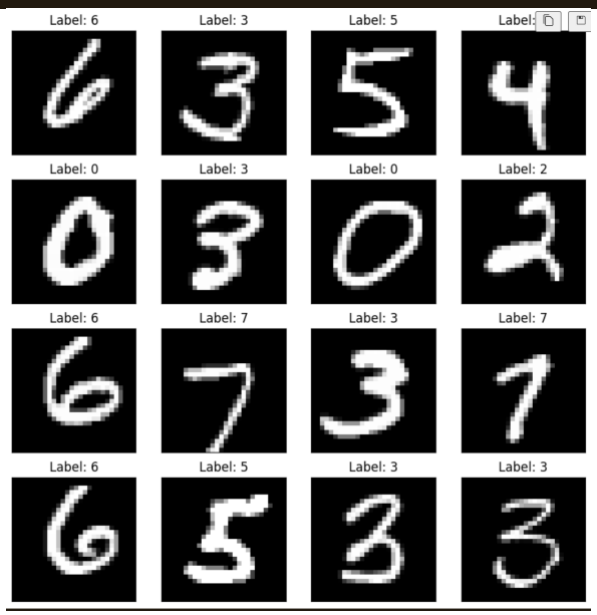
Question 2 (Total 50 points): Hand-written digits classification.

Task 1 (10 points): Dataset Preparation. In this task, you will:

1. **PyTorch Dataset Class**:

```
class MNISTDataset(Dataset):
    def __init__(self, root, transform=None):
        self.root = root
        self.transform = transform
        self.images = []
        self.labels = []
        for label in range(10):
            label_dir = os.path.join(self.root, str(label))
            if not os.path.exists(label_dir):
                continue
            for filename in os.listdir(label_dir):
                filepath = os.path.join(label_dir, filename)
                image = Image.open(filepath).convert('L')
                image_np = np.array(image, dtype=np.uint8).reshape(1, 28, 28)
                self.images.append(image_np)
                self.labels.append(label)

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, index):
        image = self.images[index]
        label = self.labels[index]
        image_2d = image.reshape(28, 28)
        image_pil = Image.fromarray(image_2d, mode='L')
        if self.transform:
            image_transformed = self.transform(image_pil)
        else:
```

```
            image_transformed = torch.from_numpy(image).float()
        return image_transformed, label

    def show_random(self):
        indices = np.random.randint(0, len(self), [16, ])
        f, ax = plt.subplots(4, 4, figsize=(10, 10))
        for i in range(4):
            for j in range(4):
                ax[i, j].imshow(self.images[indices[i * 4 + j]].reshape(28, 28), cmap='gray')
                ax[i, j].tick_params(top=False, bottom=False, left=False, right=False, labelleft=False, labelbottom=False)
                ax[i, j].set_title(f'Label: {self.labels[indices[i * 4 + j]]}')
        plt.show()

```



Pytorch Neural Network:

Implementation:

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class PytorchConvNet(nn.Module):
    def __init__(self):
        super(PytorchConvNet, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=6, kernel_size=5)
        self.conv2 = nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5)
        self.fc = nn.Linear(in_features=16 * 24 * 24, out_features=10)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.conv1(x)
        x = self.relu(x)
        x = self.conv2(x)
        x = self.relu(x)
        x = x.view(x.size(0), -1)
        x = self.fc(x)
        return x

    def name(self):
        return "PytorchConvNet Convolutional Neural Network"
```

Results:

Test loss: 0.037737, Test accuracy: 0.987900

NumPy-based Convolutional Neural Network:

Implementation is featured in code. This implementation's runtime was a massive amount slower than the PyTorch version. Took me hours for results, and I was not able to log them properly at the time, thus I was not able to attach the results. However, it is provided in the attached source code. The implementation was much more complex than the PyTorch version, especially when it came to backpropagation. I had to make many changes and add more helper functions inside the NumpyConvNet class. I also had to address padding and channel issues with the convolution because I kept getting errors. However, I got it working through condition-checking.