CV

046746

Hw4

Submitting:

Gonen Weiss – 312347982

Alexander Balabanov – 312775364

Date:

2/7/20

Q1:

Q2.0:

Given N corresponding points {p, q} we can estimate a H matrix that projects a homogeneous coordinated of qi to pi.

Observe the calculation matrix

After breaking it down we get

From the last equation we can find a value of

We will move everything to 1 side and extract the matrix coefficients

For every pair of points {p, q} we get

Let us define the equation as:

Where A is a matrix consisting of point coordinates, and h is a vector made from matrix H values.

Looking and the equation it is clear that every pair of points sets 2 constrains, with the 8 DOF that matrix H has there should be 4 linearly independent pair of points to find a solution.

Q2.1

Let us define the image that will be warped to as im1, and the image that we want to project on im1 as im2.

Taking 2 pictures as different subplots, side by side, we have implemented a manual feature matcher that enables choosing point matches between the two images in the following order: START -> Left -> Right -> Left -> Right -> … -> Left -> Right 🡪 END

The sequence is:

* Im1 – pick a feature in the image
* Im2 – find the corresponding feature

Repeat N times

Q2.2

Looking back at Q2.0, it is possible to see a method to find the matrix H, after taking the lists of corresponding points, we will create matrix A.

Matrix A will not be a square matrix, so we will take SVD approach as seen in tutorial 8.

Using SVD is beneficial in this instance, since there can be a strong noise factor on the points we had found.

Q2.3

Image wrapping should be a straightforward task, it is a per pixel function.

Taking each pixel in the input image (img) changing it into homogeneous coordinated, applying the transformation matrix and getting the new coordinated.

Unfortunately, pixel coordinated are discrete, and there might be pixels in the new image that are missing information. Those “holes” won’t be similar to “salt & papper” noise that can be removed with a median filter. They come in “paths” that correspond with the transformation so dealing with it is much harder. Similarly, the wrapped pixels do not form a square, so using an interpolation probably will not work and take longer to calculate. Therefore, we used a reverse approach, creating a blank wrapped (wrap) image and working backwards. Taking each pixel coordinates in wrap and checking where it should be on the original (img), using a reverse transform of H. The resulting coordinates are checked if in bounds and interpolated with img. This method insures getting a continuous wrapped image.

Q2.4

Panorama stitching includes 2 main actions, finding a mask for both images and blending nicely. Since img1 was not wrapped it contains less interpolation artifacts so we will use all of it. Wrap\_img2 needs to be masked to the information that does not overlap with img1 and added to the image.

Blending will be discussed in Q2.9

Q2.5

Autonomous stitching can be as good as it’s key point extractor and matcher. First we tried using ORB to get key points (KP) and descriptors (DS) but it didn’t work well, so we used a SIFT algorithm from git hub as was recommended (<https://github.com/rmislam/PythonSIFT>).

SIFT works similarly to what we had implemented on Hw1, finding edges and corners with difference of Gaussian pyramid (DoG pyramid), using Sobel filter (differentials) to filter out edges and leave most corners, and finally using an anker filter to get a numerical descriptor of the corner.

Once both images are calculated we need to find matching KP, likely we have DS that should be mostly invariant, the matcher will take each both lists of KP, DS and match them one by one, therefore it is called brute force.

Not all matches are great so we will want to find as many good matches and no outliners. Distance can be really misleading measurement, even if we use KNN to have a estimation how sure is it. Therefore, the only manipulation we will do is to sort them by distance and use RANSAC as described in Q2.8

Q2.6

Blank

Q2.7

First we will run both algorithms manually and automatically on both image sets, and compare the results

As stated in Q2.5 stitching can be as good as it’s KP finder, obviously a human can do the task better and the results are visibly better, however human can’t go through huge data sets and that is the manual method downfall. Manual

Q2.8

RANSAC’s advantage is not the random factor, but the iterative nature. Lists p1, p2 are sorted by distance, we will tap into it by using a weighted randomization. We will give higher probability for KP in the head of the lists, thus increasing the chance to find the few good pairs that include most other pairs. We picked 8 points at a time to filter out noise in the data.

Q3.1