COMP-558 Assignment #4

Zhaoqi Xu
260563752

Question 1: SIFT Features

This paper introduced an approach named the Scale Invariant Feature Transform (SIFT), as it transforms image data into scale-invariant coordinates relative to local features. There are mainly four steps involved in SIFT algorithm, scale-space extrema detection, accurate keypoint localization, orientation assignment and keypoint descriptor. We will give a brief discussion on each step.

1.Scale-space extrema detection. The author used scale-space filtering to detect locations that are invariant to scale change of the image. Under a variety of reasonable assumptions, the only possible scale-space kernel is the Gaussian function. Therefore, the scale space of an image is defined as a function (1),

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1}$$

where $*$ is the convolution operation, $I(x, y)$ is the image and $G(x, y, \sigma)$ is a variable-scale Gaussian. To efficiently detect stable keypoint locations in scale space, they used scale-space extrema in the difference-of-Gaussian (DOG) function (2),

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2}$$

Furthermore, DOG gives an approximation to the scale-normalized Laplacian of Gaussian, and the normalization of Laplacian with the factor σ^2 is required for true scale invariance. The relationship between DOG and scale-normalized Laplacian of Gaussian is:

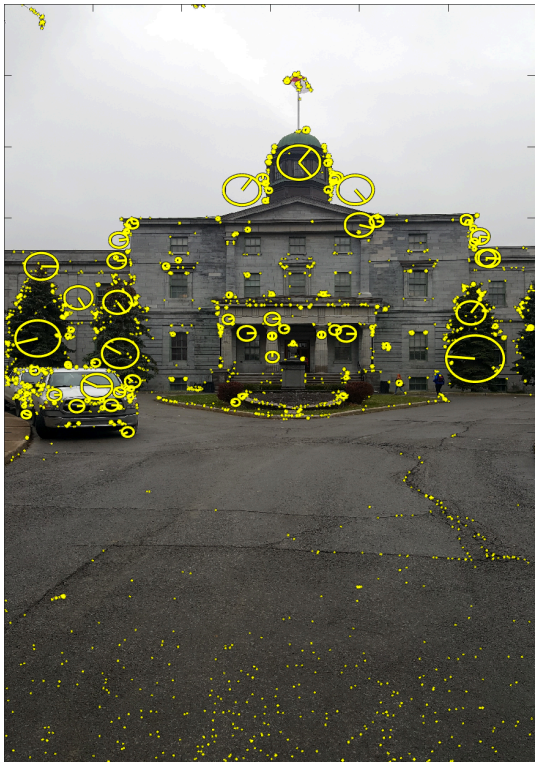
$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 G$$

Hence, scale-invariant property is achieved. Once these DOGs are found, images are searched for local extrema over scale and space. In order to detect the local maxima and minima of $D(x, y, \sigma)$, each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below. If it is a local extremum, it is a potential keypoint. It basically means that keypoint is best presented in that scale.

2.Accurate keypoint localization. Once potential keypoints location are found, they have to be refined to get more accurate results. Their approach uses the Taylor expansion of the scale-space function to get more accurate location of extrema. The location of extremum, $\hat{x}$, is determined by taking the derivative of this function with respect to x and setting it to zero. The intensity at this extremum is less than a threshold value (0.03 in paper), it is rejected. However, for stability, it is not sufficient to reject keypoints with low contrast. DOG has higher response for edges, so edges also need to be removed. For this, they used a 2x2 Hessian matrix to compute the principal curvature. If the ratio between the largest magnitude eigenvalue and the smaller one is greater than a threshold (10 in paper), that keypoint is discarded.

3. Orientation assignment. Now they assign a consistent orientation to each keypoint based on local image properties to achieve invariance to image rotation. They first take a neighborhood around the keypoint location depending on the scale, and calculate the gradient in that region. An orientation histogram with 36 bins covering 360 degrees is created. The highest peak in the histogram is taken and any peak above 80% of it is also considered to calculate the orientation. Therefore, it creates keypoints with different directions but the same scale and location.
4.Keypoint Descriptor. Last, they create keypoint descriptor. A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, which are then accumulated into orientation histograms summarizing the contents over 4x4 sub-regions with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. In this paper, a 16x16 neighborhood around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histograms are created. So, a total of 128 bin values are available.

The yellow circles in the images below illustrate the keypoints extracted by SIFT algorithm (vl_sift). The peak threshold is 7 and the edge threshold is 10.



Question 2: RANSAC

The RANSAC algorithm works for a general model fitting problem by repeating the following steps:

1. Randomly select a sample subset containing minimal data items from the original data. Call this subset the hypothetical inliers.

2. Compute a fitting model and the corresponding model parameters using only the elements of this sample subset.
3. All other data are then tested against the fitted model. Those points that fit the estimated model well, according to some model-specific loss function, are considered as part of the consensus set.
4. The estimated model is reasonably good if sufficiently many points have been classified as part of the consensus set.
5. Once a reasonably good estimated model is selected, it may be improved by re-estimating it by using all members of the consensus set.

Use RANSAC to solve for the homography between two images:

1. Find feature points (SIFT features) in each of the two images: $\{(x_i, y_i)\}$ and $\{(x_i', y_i')\}$.
2. For each feature point (x_i, y_i) in one image, we use the keypoint descriptors to find a candidate feature point (x_i', y_i') in the second image whose intensity neighborhood is similar to (x_i, y_i).
3. Step 2 gives a set of 4-tuples (x_i, y_i, x_i', y_i'). Randomly choose four 4-tuples and fit an homography that correctly maps the four $\{(x_i, y_i)\}$ to their corresponding $\{(x_i', y_i')\}$. Find other 4-tuples (consensus set) for whose the distance of the 4-tuple from the model is sufficiently small.
4. Choose the homography which yielded the largest consensus set by repeating step 3 a certain number of times.
5. Re-estimate the homography using least squares within the consensus set.

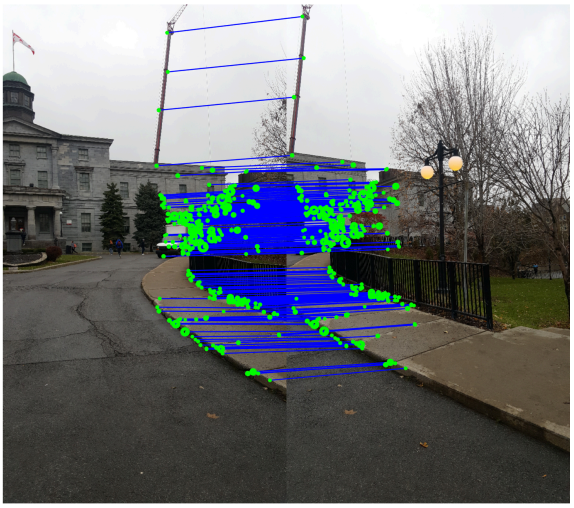
Question 3: Feature Matching

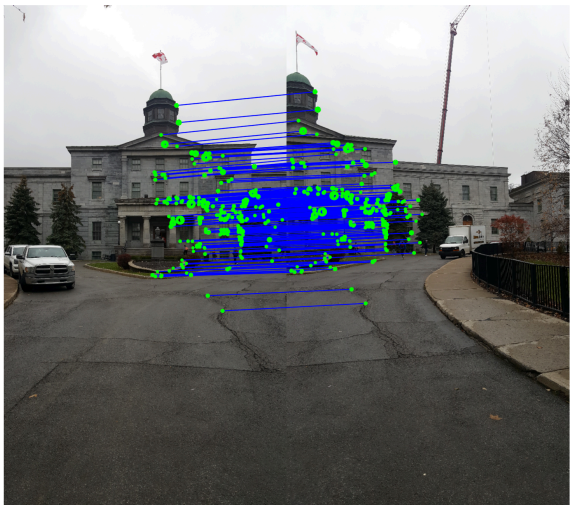
Instead of using code provided by professor, I used the VLFeat open source library in this question. To select the SIFT features in images, I used a built-in function vl_sift(Image, peak_thresh, edge_thresh) where peak threshold and edge threshold can be explicitly chosen. The vl_sift function computes the SIFT keypoints and corresponding descriptors using the algorithm we have seen in D. Lowe's paper.

I noticed that the SIFT detector is controlled mainly by two parameters: the peak threshold and the edge threshold. As the peak threshold increases, the number of selected SIFT features decreases. While as the edge threshold increases, the number of selected SIFT features increases as well. After parameter tuning, I ended up with using 4 and 10 for peak threshold and edge threshold respectively and got a reasonable number of SIFT features to perform feature matching.

Then I used another built-in function in VLFeat vl_ubcmatch(descriptor1, descriptor2) to matches the two sets of SIFT descriptors. The function uses the algorithm suggested by D. Lowe to reject matches that are two ambiguous. The strategy behind this algorithm is as follows: the best candidate match for each keypoint is found by identifying its nearest neighbor in the set of keypoints from the other image. The nearest neighbor is defined as the keypoint with minimum Euclidean distance for the invariant descriptor vector. However, in some cases, the second closest-match may be very near to the first due to noise or some other reasons. In that case, the ratio of closest-distance to second-closest distance is taken. If it is greater than 0.8, they are rejected. This strategy eliminates around 90% of false matches while discards only 5%

correct matches, as per the paper. The two images below illustrate the feature matching results using this algorithm.





Question 4: Use of RANSAC to Stich Images

I implemented RANSAC algorithm as question 2 based on the results of question 3. Firstly, I randomly selected 4 pairs of feature matchings and computed the homography of these four matched points by first doing a direct linear transform (DLT) and then applying SVD on it.

$$UDV^T = A$$

Then homography H is obtained by reshaping V. To compute the score of each homograph, I first applied the homography on matched keypoints of image A, then calculated the distance between the matched keypoints of image B and the matched keypoints of image A transformed by applying homography. If the distance is smaller than a threshold (35 in my implementation), then this keypoint is considered as an inlier. The number of the inliers is the homography's score. In my implementation, I repeated above process 100 iterations, namely, 100 homographies were found. But we are only interested in the one with the largest consensus set. Then I re-estimated the homography with highest score with the consensus set.

To restrict both images in the same grid after transformation, I use the homography to map four corners of the second image to the pixel coordinate frame of the first image. Now the remaining part is to stitch these two images. To do this, I apply the homography directly to all pixels in one of the image and draw those intensities in another image. If a scene point is not defined in one image, we set its intensity at this point as zero. Then the intensity at this scene in panorama is the intensity value of the other image. If a scene point is defined in both image, then the final intensity value is the average of them. If a scene point is not defined on both images, then the intensity value is zero, this is where the pure black pixels come from. In this way, overlap is handled.

Two images below illustrate the panorama for each data set. Planar projection is used in my implementation. Actually, the algorithm worked pretty well on these two image sequences. However due to the memory limitation, I can only generate the panorama with three consecutive images. Since we are not asked to use any blending techniques, the intensity

difference of overlapping pixels can be observed. Zoom in the image, I also observed ghosting on people, flags or trees. I think this ghosting is caused by motion in the scene. People were moving when these pictures were taken, and the wind was blowing the flags and trees. Also, re-estimating the homography with all consensus set is very important to improve the stitching performance which is very straightforward to understand. The more inlier keypoints we use to estimate the model, the better model we get to represent the data.

Question 5: Failure of Algorithm

To make the algorithm fail, I came up with a set of images as follows.

1

2

3

Then I tried to stitch image 1 and 2, the panorama is as follows:

Though there are many co-visible objects in image 1 and 2, the algorithm still is not able to stich them perfectly. The background of the image was perfectly stitched but the car was not. The reason behind is that though the SIFT is rotation invariance, it works well only if correct main orientation is found consistently among different views. If the orientation of two images is quite different, then the image stitching algorithm rely on pairwise homographies would fail. It turns out that large view point changes cause the homography cannot transform all matching features from one image to the other. An extreme case is that when I tried to stitch image 1 and 3, it was even not able to find an appropriate homography mapping one to the other.