Computer Vision

Local Features

Student: Yuchang Jiang

Date: 8 Oct. 2020

1. Detection

1.1 Implementation Process

For detection purpose, Harris corner detector is implemented in $extractHarris.m$. The first step is to compute image gradient in x and y direction based on convolution (like [0.5, 0, -0.5]). Then for local auto-correlation matrix, compute $I_{xx} = I_x \cdot I_x$, $I_{yy} = I_y \cdot I_y$, $I_{xy} = I_{yx} = I_x \cdot I_y$ and add Gaussian weights by $imgaussfilt$ function. When combining $I_{xx}, I_{yy}, I_{xy}, I_{yx}$ for each pixel, local auto-correlation matrix, M_p for each pixel is gained. The third step is to compute Harris corner response function, which is defined as $C_{i,j} = det(M_{i,j} - kTr^2(M_{i,j}))$ and k is an empirical constant here. After obtaining $C_{i,j}$ for each pixel, two detection conditions are defined to filter out corner points:

- $C_{i,j}$ should be above a defined threshold, t
  To meet this condition, any $C_{i,j}$ smaller than threshold is removed
- $C_{i,j}$ is a local maximum in its 3x3 neighborhood
  To meet this condition, $imregionalmax$ is used to find maximum point in 3x3 region

1.2 Result and Discussion

Completing this detection process, keypoints are extracted. However, there are several parameters to tune: sigma for Gaussian, k in corner response function and t, the threshold to filter out keypoints. To check how each parameter affect the final detection result, experiments with $sigma = 0.5, 1, 2$, $k = 4e-2, 6e-2$, $t = 1e-5, 1e-4$ are performed and results are shown in Figure 1 and Figure 2. For $t = 1e-6$, experiment is tried but this small threshold leads to a very large number of keypoints, which is not useful so related experiment for $t = 1e-6$ is not displayed here.

Based on those results, the influence of each parameters can be observed:

- For sigma: it seems more corners would be detected when it is 1, than it is 0.5 or 2. Besides, when sigma is 0.5 and 1, the edge points between the table and background wall are detected. Thus, a good choice for our case is sigma = 2.
- For k, if k is larger, the corner response is smaller, fewer corners will be detected.
- For t, if t is larger, fewer corners will be detected.

Based on visual observation for experiment results, $sigma = 2, k = 4e-2, t = 1e-5$ are chosen for later process.

Figure 1: Experiment result for image 1

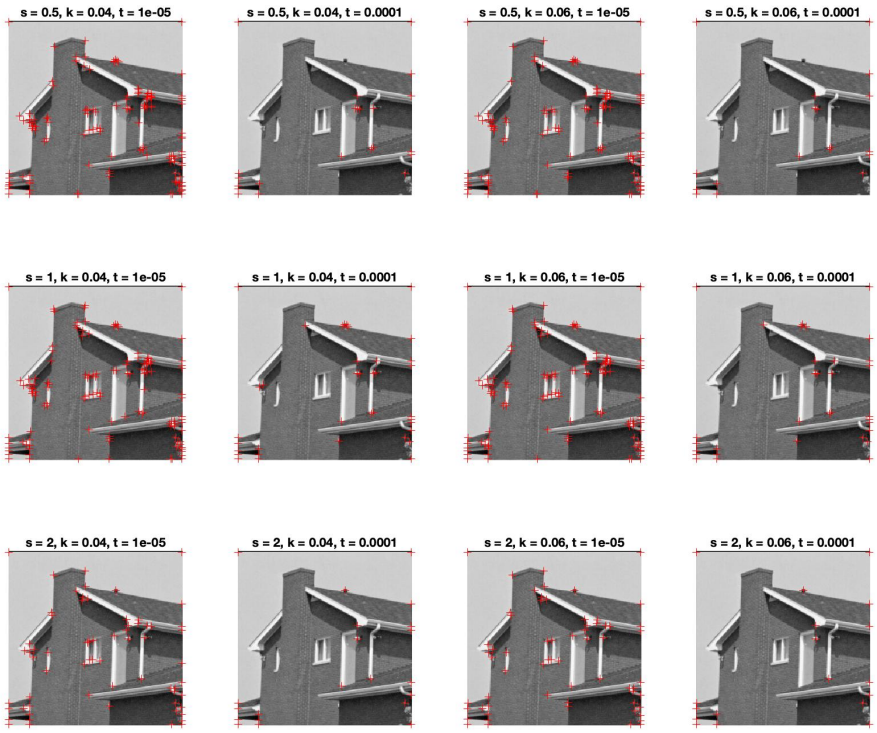

Figure 2: Experiment result for image 2

2. Description & Matching

2.1 Implementation Process

The first step for descriptor is to remove border points: as observed in Figure 1 and 2, border points are detected as corners, which is annoying. Thus, all detected corners within a define width near borderline is removed. (the width used is 5 pixel here). Then patches with 9x9 size is extracted for each corner point.

After extracting patches in $extractDescriptors.m$, $matchDescriptors.m$ is implemented to calculate Euclidean distance between patches in image 1 and image 2. Here $pdist2$ function is employed to compute distance between descriptors (or patches). Now computed distance is saved in a matrix (mxn) if there are m keypoints for image 1 and n keypoints for image 2 so $distance(i, j)$ corresponds to the pairwise distance between patch i in image 1 and patch j in image 2. Then 3 match protocols: one-way, mutual and

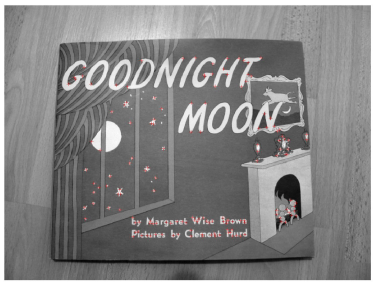

Figure 3: Keypoints in image 1

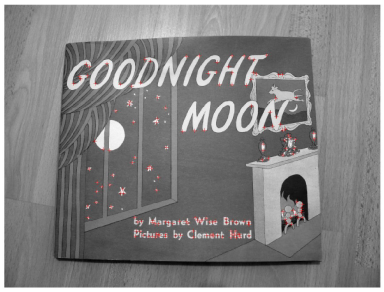

Figure 4: Keypoints in image 2

ratio are used to find matches:

- One-way matching protocol
  At first, for each row of distance matrix, find the minimal number, which means find the nearest neighbor in image 2 for each keypoint in image 1 (so function $[m, idx] = min(distances, [], 2)$ is used. Thus, the number of resulted matches equals to the number of keypoints in image 1.

- Mutual matching protocol
  Based on one-way protocol, mutual protocol can ensure the mutual correspondence for keypoints in two images. Besides computing the nearest neighbor in image 2 for each keypoint in image 1, the nearest neighbor in image 1 for each keypoint in image 2 is also computed ($[m, idx2] = min(distances, [], 1)$). Then comparing two resulted match matrix to filter out same matches in two matrices, which means swapping images will lead to same matches.

- Ratio matching protocol
  One-way protocol only cares about the first minimal distance between matches. Here the nearest neighbor with second minimal distance is also calculated with $mink(distances, 2, 2)$. Then a ratio threshold 0.5, 0.6, 0.8 are tried and only matches with $\frac{first\ minimal\ distances}{second\ minimal\ distances} < ratio\ threshold$ are kept.

2.2 Result and Discussion

As shown in Figure 3 and Figure 4, there are 152 and 168 keypoints in image 1 and image 2, separately.

As shown in Figure 5, there are 152 matches in one-way matching protocol, the same as the number of keypoints in image 1. Some matches with non-parallel lines are obviously wrong. Thus, a better protocol is needed to improve matching correctness.

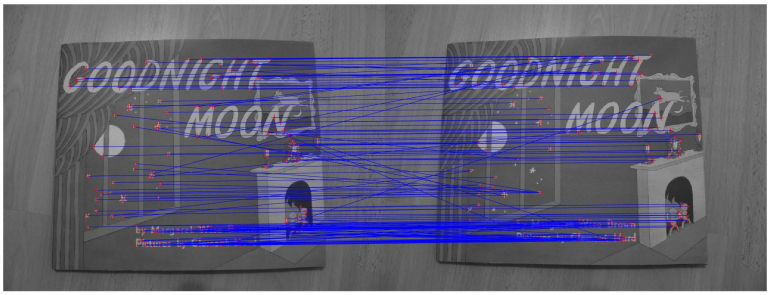

Figure 5: Result for one-way matching protocol

For mutual matching protocol, it produces 106 matches, as shown in Figure 6. There are fewer matches than the result of one-way matches and there is one obvious non-parallel line, which is wrong matching. Generally, the quality of mutual protocol is better than one-way protocol but mutual protocol still produces obvious wrong match.

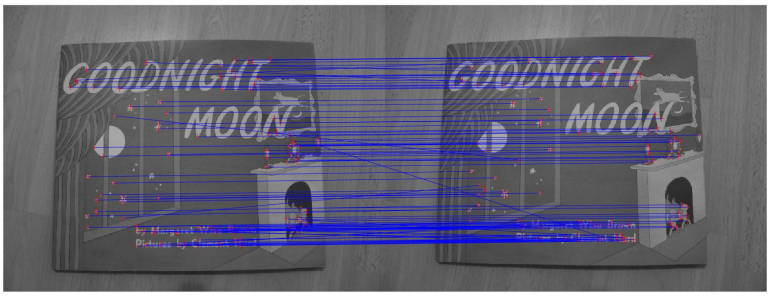

Figure 6: Result for mutual matching protocol

When using ratio protocol, the different ratio thresholds will lead to different number

of matches in the result:

- as shown in Figure 7, when choosing 0.8 ratio threshold, there are 94 matches, including one incorrect non-parallel matching line.
- as shown in Figure 8, when choosing 0.6 ratio threshold, there are 61 matches and no obvious incorrect matches are observed.
- as shown in Figure 9, when choosing 0.5 ratio threshold, there are 44 matches and no obvious incorrect matches are observed.

To summarize, a larger ratio threshold leads to a larger number of matches in the result but more possible to contain wrong matches so there is a balance between false positive and false negative. Setting the ratio too high, like 0.8 can lead to many incorrect matches (high false positive) but very low ratio will leads to a few matches (high false negative), Thus, to balance this effect, choosing 0.6 threshold is probably better.

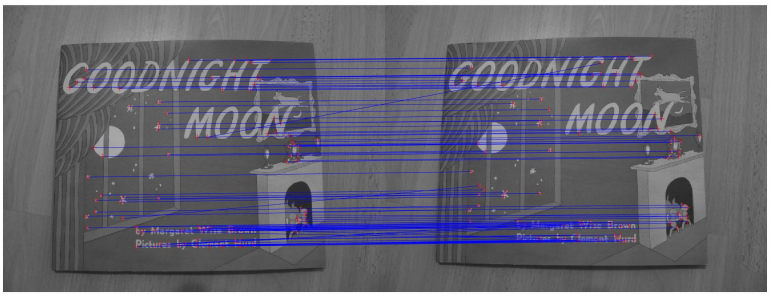

Figure 7: Result for ratio matching protocol (threshold: 0.8)

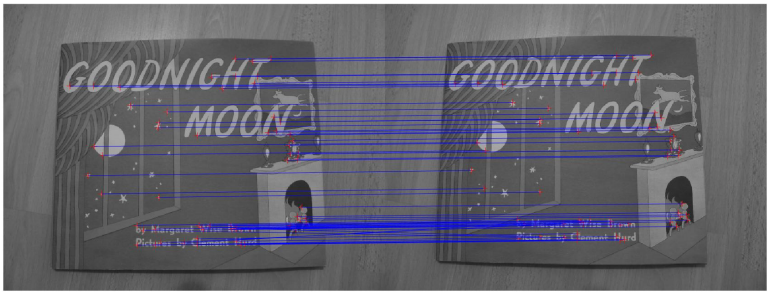

Figure 8: Result for ratio matching protocol (threshold: 0.6)

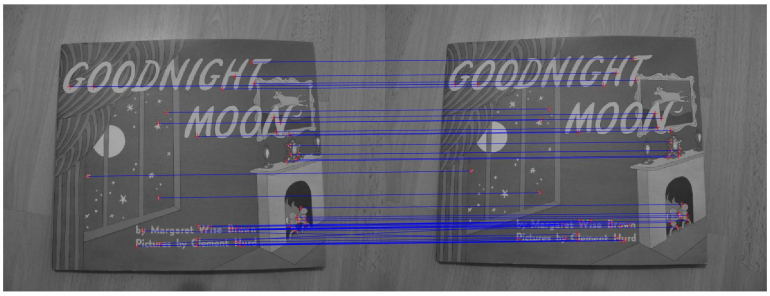

Figure 9: Result for ratio matching protocol (threshold: 0.5)