

Problem 1 Homography Estimation

(a) Perform local feature detection (SIFT) on each image


(b) Q1-1 Find the correspondence between anchor image and target images by descriptor matching.

Reject the outliers by ratio test or manual comparison. Select top k pairs from the matching result, where $k = 4, 8, 20$. Briefly explain how do you reject the outliers (i.e. mismatching) and visualize your matching results for each k value.



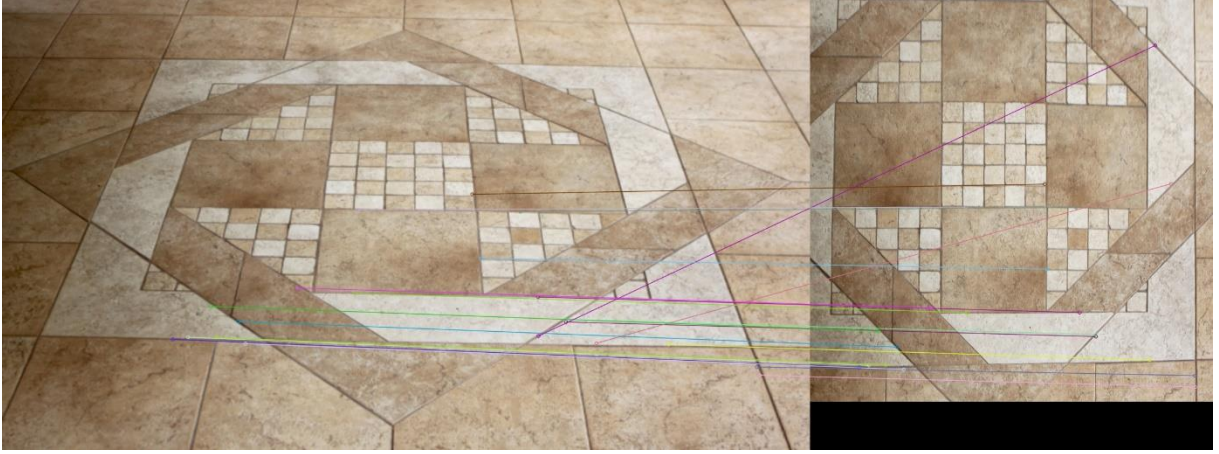
To reject the outliers, I perform:

1. Ratio test: First use `matcher.knnMatch(k=2)` to find most matching pairs for each point. Only when the distance of the most dominant feature pairs is smaller than $0.4 \times \text{distance of the second dominant feature pairs}$, can this most dominant pairs can be view as good matches. (The matching result I shown in Table 1 and Table 2)

(Table 1) Matching Result for pair1(1-0.png, 1-1.png)

k	
4	
8	
20	

(Table 2) Matching Result for pair2(1-0.png, 1-2.png)

k	
4	
8	
20	

However, this test could still cause mismatching, I deigned another manual filtering to eliminate outliers.

2. Manual Comparison: Relative Position Comparison

In these cases, I assume both transformations do not have dramatic variation on horizontal viewpoint changes, but mainly transform on vertical(z-axis) viewpoint. As a result, for each corresponding pair, both of its points should locate at similar relative region of their own image. I observe from the wrong matching below. In fig.1, the highlighted corresponding pairs. They are mismatched and belongs to different relative regions. I believe eliminate them can get better result.

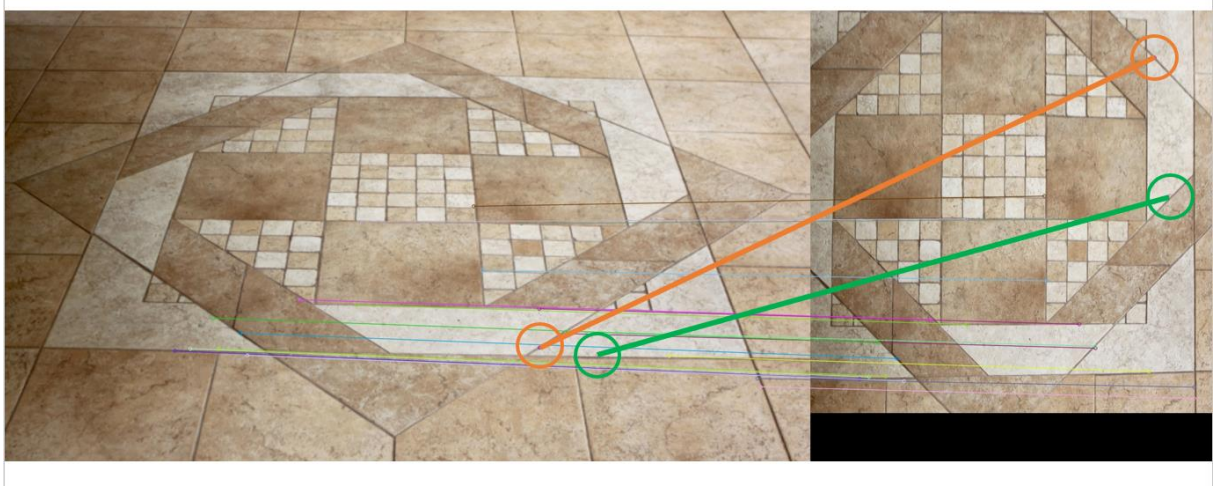


Fig.1

So, I separate each image into 3*3 regions separately (show in fig.2) and calculate which region each point is located.

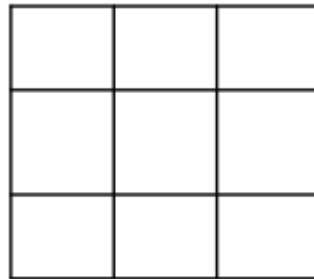


fig.2

If any found good matches, its two corresponding points locate at different relative region. I would remove this pair from good matches set.

Table 5 shows how this design improve the matching.

Q1-2 For each k value, estimate the homography between anchor image and target images with direct linear transform. Compute the reprojection error with the ground truth matching pairs

(Table 3)

k	Reprojection error	
	Pair1 (w/o Relative Position Comparison)	Pair2 (w/ Relative Position Comparison)
4	138.82	344.97
8	1.5	12.68
20	0.29	3.88

The more corresponding pairs are used, the more accurate the result is.

Q1-3 Similar to (c), but use normalized direct linear transform instead. Compute the reprojection error and compare the results with Q1-2.

(Table 4)

k	Reprojection error			
	Pair 1(w/o Relative Position Comparison)		Pair 2 (w/ Relative Position Comparison)	
	w/o normalized	w/ normalized	w/o normalized	w/ normalized

4	138.82	138.82	344.97	344.97
8	1.5	1.44	12.68	29.6
20	0.29	0.28	3.88	2.28

Method: I normalized points of equal image, shifted them to mean value = (0, 0), average distance = $\sqrt{2}$
Normalization improves the result only a little bit. Perhaps it's because the scale of images are not dramatically different.

Experiment

I did some experiments to prove my design works. In Table 5, we can observe pair2 transformation has really bad matching. The "Relative Position Comparison" method really helps getting better matching in this task. However, it does not help in pair 1 transformation.

(Table 5)

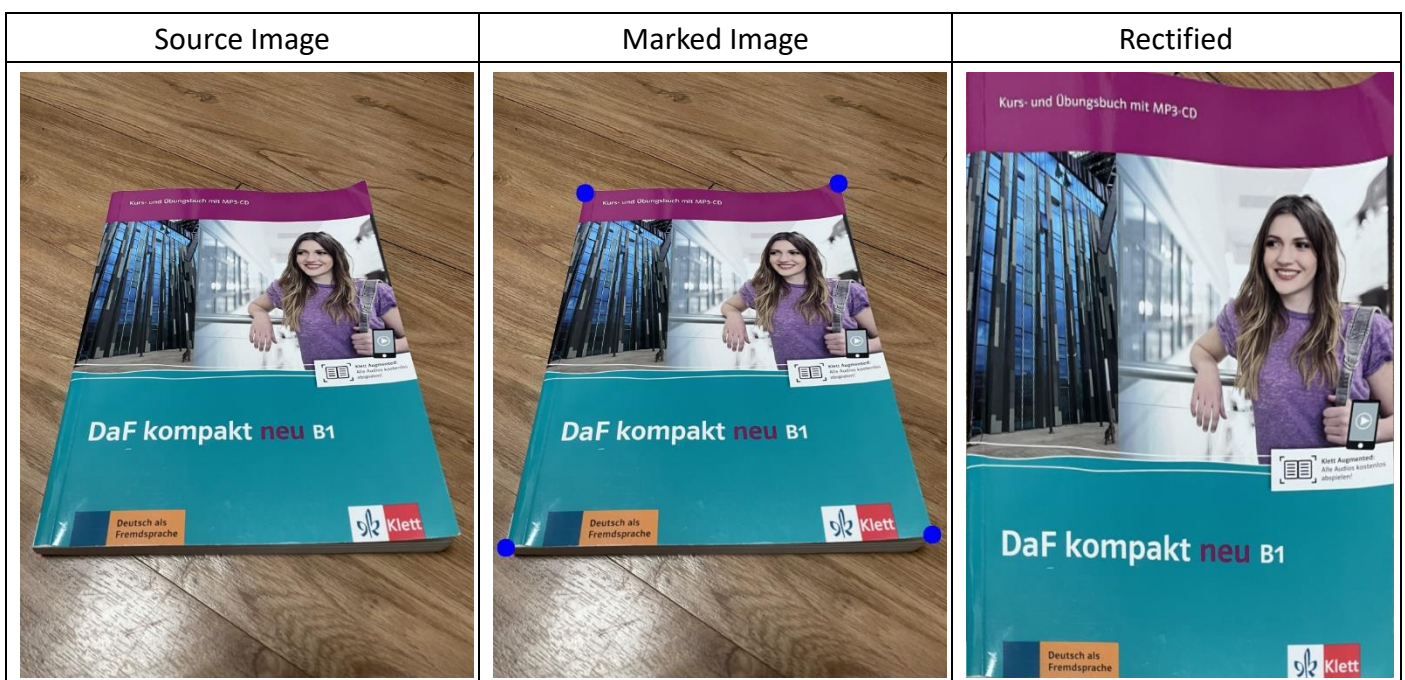
k	Reprojection error			
	Pair 1 (w/o normalized)		Pair 2 (w/o normalized)	
	w/ RPC	w/o RPC	w/ RPC	w/o RPC
4	2072.22	138.82	344.97	344.97
8	1.8	1.5	12.68	530.82
20	0.29	0.29	3.88	552

(RPV = Relative Position Comparison)

Discussion

SIFT matching is not an accurate method. It requires lots of extra processing. Perhaps there're some deep learning techniques to fulfill this task. However, it's still an important concept for 3D reconstruction. I'm glad to have this practice to help me understand the Homography.

Problem 2 Document Rectification



Discussion:

The result seemed to have little distortion, I think it's caused by the rough surface of the cover. And this roughness was transformed and amplified.

Method:

1. Corners

I chose the 4 corners of a book as source points and want to project them to 4 corners of a rectangle (A4 size). I used *setMouseCallback* tool to detect the position of corner.

2. Warping

I formulate backward warping. The steps are as following:

- Get target image shape (W_t , H_t) and create a 2D grid to store the position of each pixel in target image.
- Multiply inverse H on target position to know the corresponding points in original image.
- Use *cv2.remap* to transform the image.

Code

Language: Python

Package requirements: Numpy, Opencv, math

To execute my codes

Problem 1:

```
python3 1.py <source image> <target image> <gt_correspondence.npy>
```

ex: `python3 1.py images/1-0.png images/1-1.png groundtruth_correspondences/correspondence_01.npy`

This will output reprojection error value.

Parameter:

- **pairs_num:** how many pairs of corresponding points you want to use
- **normalize:** normalized DLT or DLT

Problem 2:

```
python3 1.py <input image>
```

ex: `python3 2.py lr_p2.png`

This will generate the rectified output from `lr_p2.png`.