Lab05: Feature Descriptor and Correspondences

Oct 2022

Problem 1

Some of the "optimal" results with chose parameters:

		Planar rotation	Views	Scale and planar	illumination
SSD	Window	7	3	7	7
	Bins	0.2	0.5	0.4	0.5
	Accuracy	0.8793	0.0341	0.0588	0.2581
NCC	Window	7		15	7
	Bins	0.95		0.99	0.8
	Accuracy	0.8333	/	0.0054	0.9767
Chi-Square	Window	21			21
	Bins	0.7			0.45
	Accuracy	0.2500	/	/	0.8571
SIFT	Accuracy	0.9883	0.1754	0.7870	0.9504

*the ratio for NCC, which we select bigger number to show, I make the ratio multiply with simi first, so it still below 1.

*The bins is set to 10 in the above tests. I've tried bigger or smaller ones with Chi-Square.

One of the criteria is to keep successful match over 1, so the result accuracy won't be 1/N, or 2/N.

Basically, when then accuracy is fine, then the higher the Lowe's test ratio is, the lower the accuracy will be, it may not be the case in some of the methods.

The windows size has to be tuned several times for the best result.

There are chances for Chi-square and NCC to come out with 0 accuracy in certain kind of picture like pic 2&3. Need to mention that it could be my code's flaw.

For SSD. The accuracy is unstable, it works fine with planar rotation and illumination change, but other 2.

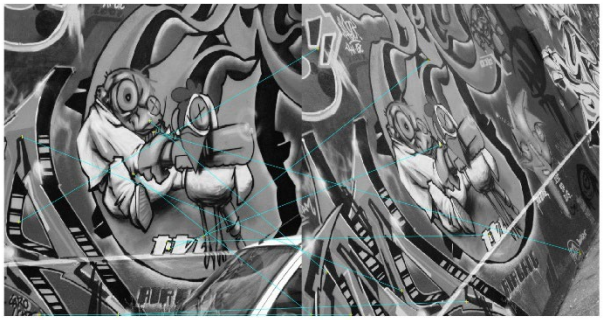
For NCC. It works well with planar rotation and illumination change, but awful in other two.

For Chi-square method. With higher bins size (like 16), the accuracy could rise a bit, but if the bins=32, then the accuracy drops. It works well with illumination changes, and performs bad in other three changes comparing to other methods.

To sum up, the SIFT outcompetes the three methods above with better accuracy and uses less time.

The NCC performs better than other two, since the P1 P4 scenarios are very suitable for this method. Though the SSD has outputs for each scenario, the number of pairs are pretty small that can't be used to get H. Obviously, these methods can't handle scenarios such as Views or Scale and planar changes.

SSD:



NCC:

Chi:Square:

SIFT:

Problem 2:

The result of two improvement for Problem 1:

		Planar rotation		Views		Scale and planar		illumination	
		Bi	Uni	BI	Uni	Bi	Uni	BI	Uni
SSD	Window	7		3		7		7	
	Bins	0.2		0.5		0.4		0.5	
	Accuracy	0.8919	0.8947	0.0741	0.0423	0	0.588	0.3333	0.2727
	P1	0.8793		0.0341		0.0588		0.2581	
NCC	Window	7				15		7	
	Bins	0.95				0.99		0.8	
	Accuracy	0.8739	0.8415	/	/	0.0741	/	1	0.8672
	P1	0.8333		/		0.0054		0.9767	
Chi-Square	Window	21						21	
	Bins	0.7						0.45	
	Accuracy	0.3333	0.2500	0.01	0.01	/	0.0109	1	0.8571
	P1	0.2500		/		/		0.8571	

*In order to compare with Problem1 results, I used identical parameters.

It turns out that the accuracy can be improved, but not significantly. Since both improvements are just delete some of the noise and mismatched pairs, no more matches are found, thus they can't make these three methods better than sift.

Problem 3:

Process:

Step1: Filter the image with Gaussian filter to reduce the noise (e.g. sigma=2 N=9)

Step2: Take the feature as center, in a window with size=[S,S] attached to the center. Randomly picking two pixels and compare their intensities, and assigning them with binary numbers as the function below.

$$\tau(\mathbf{p};\mathbf{x},\mathbf{y}) := \begin{cases} 1 & \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

Step3: Randomly picking N pairs of compared pixels and execute the Step2, and defines a set of binary string. According to the author, both X and Y ~ i.i.d. $Gaussian(0, \frac{S^2}{25})$, this method performs best.

Step4: After the steps above, we have binary string of which length is 256 bits for every feature. And matching features with Hamming distance instead of Euclidean Distance, and two criteria of the judgement are as below:

1. Two features' binary strings share less than 128 same corresponding numbers won't be taken as matched features.
2. The feature matches with the feature in other images who has the most common digits in binary strings.

With the steps, the features can be matched with BRIEF descriptor.

Why it is faster depends on the simple description for the feature, it just needs N times of comparison. Also, the Hamming Distance is way more efficient under this case than Euclidean Distance since it just need to count the common digits in binary strings instead of multiplies and division.