Homework 2 Writeup

Instructions

- This write-up is intended to be 'light'; its function is to help us grade your work.
- Please describe any interesting or non-standard decisions you made in writing your algorithm.
- Show your results and discuss any interesting findings.
- List any extra credit implementation and its results.
- Feel free to include code snippets, images, and equations.
- Use as many pages as you need, but err on the short side.
- **Please make this document anonymous.**

Assignment Overview

My project is a multi-functional program that matches the same features in images of same object taken in different angles. Specifically, it has three parts: detecting the corners, featurizing the points, and finally matching the corresponding features. There are two core parts to this project. One is the Harris corner detector's equation that uses the second moment matrix to calculate the Harris Cornerness Score and the other is the SIFT feature matching equation. See Equation 1, 2.

$$C = det(M) - *trace(M)^2 \tag{1}$$

$$D = sqrt(A - B) \tag{2}$$

Implementation Detail

My project follows the steps described in the assignment overview to implement the mathematical process to find corners, get features, and match features. Specifically, I used np.gradients and feature.peaklocalmax() to obtain corners. For the feature creating, I had to use a nested for loop to implement creating histograms since I wasn't allowed to use np.digitize. For matching the features, I used np.matmul() as a core part of the code.

My code snippet highlights an interesting point.

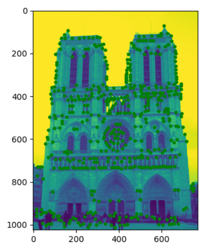
```
#Calculate Harris Response
det = Ixx * Iyy - Ixy ** 2
trace = Ixx + Iyy
harris_response = det - alpha * trace ** 2

#feature matching
f1 = np.sum(np.square(im1_features), axis=1, keepdims=True)
f2 = np.sum(np.square(im2_features), axis=1, keepdims=True).transpose
                                    ()

A = f1 + f2
B = 2 * np.matmul(im1_features, im2_features.transpose())
D = np.sqrt(A - B)
```

Result

1. Result 1 was a total failure, because I did not set parameters for peak local max, so I got an image covered with dots.
2. Result 2 (Figure 1, left) was surprising, because I saw that corners were detected correctly.
3. Result 3 (Figure 1, right) blew my socks off, because the corresponding corners were fairly correctly matched.



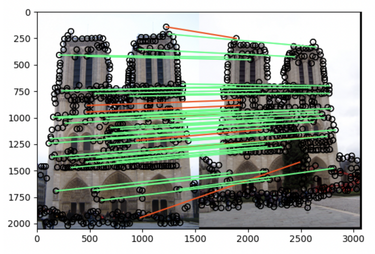


Figure 1: *Left:* Corner Detecting *Right:* Feature Matching

My results are summarized in Table 1.

N Most Confident Matches Test	Accuracy
50 matches	84%
100 matches	69%

Table 1: Stunning revelation about the efficiency of my code.

Extra Credit (Optional)

1. Implementation A, code snippets, and results

```
one = 1;
two = one + one;
if two == 2
    disp( 'This computer is not broken.' );
end
```

2. Implementation B, code snippets, and results

```
one = 1;
two = one + one;
if two == 2
    disp( 'This computer is not broken.' );
end
```