Problem Set 5: Harris, SIFT, RANSAC

Supplemental Document

This short document is to help you understand the relevant API for using SIFT libraries for PS5. This is in no way the definitive guide, such documentation already exists, instead it is designed to help you quickly identify which functions you will need and which ones you can likely ignore. We give relevant pointers for OpenCV-Python.

1. OpenCV - Python

OpenCV has a non-free SIFT implementation in Python. You can find a brief tutorial here: http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html#sift-intro.

a. **Python OpenCV installation** (try to get OpenCV 2.4.9+):

Windows users check out:
http://docs.opencv.org/trunk/doc/py_tutorials/py_setup/py_setup_in_windows/py_setup_in_windows.html

Linux users try :

```
sudo apt-get install python-opencv
```

Mac users, the easy way is to use Homebrew to install OpenCV and manually setup the path.

b. **Feature Extraction:**

The main classes you will be using are the SIFT classes documented above.You will be using `cv2.KeyPoint` class to define the keypoint location of your Harris corners for input to the SIFT feature extractor.

For each detected Harris keypoint you will create a `cv2.KeyPoint` instance, where you set the values of `x` and `y` appropriately to the corner location, `_size` to the size of the local region, and the value of `_angle` to the dominant orientation computed for the corner. Additionally you will set the value of `_octave` to 0, since all points were located at the full scale version of the image. Here is an example.

```
keypoint = cv2.KeyPoint(x=10, y=10, _size=3, _angle=90, _octave=0)
```

You will need to put the `cv2.KeyPoint` instances for all of your data into a list.
You will then need to create an instance of the `class cv2.SIFT`. This is simply done by:

```
sift = cv2.SIFT()
```

Extracting the SIFT descriptors then requires you to call the `SIFT.compute()` function:

```
keypoints, descriptors = sift.compute(I, keypoints)
```

Here `I` is the original input image to compute the descriptors of, `keypoints` is the list of keypoints. It returns the descriptors for the points in `descriptors`, a `NumPy array` with the i^{th} row corresponds to the 128-element SIFT feature extracted at the location of `keypoints[i]`.

c. **Feature Matching:**

To find the putative matches you will use the class `cv2.BFMatcher`.

You will first need to create an instance of this class:

```
bfm = cv2.BFMatcher()
```

You can then compute matches for descriptors `desc1` and `desc2` as:

```
matches = bfm.match(desc1, desc2)
```

This returns a list of `cv2.DMatch` objects.

For a given match, the keypoint index from descriptor set 1 will be the value in `dmatch.queryIdx` and that from set 2 will be at `dmatch.trainIdx`. If you are interested, it is also easy to implement the ratio test in the original SIFT paper. Here is an example: http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html#matcher