Project # 2: Local Feature Extraction, Detection and Matching

CSC 391: Introduction to Computer Vision

Instructor: V. Paúl Pauca

due date: Friday, March 1st

---

1 Edges and Corners in Video

Programming part. Implement Python code that uses your laptop's camera to capture video, extracts edges and corners from each video frame, and superimposes these features back on the frame for visualization. You do not need to program edge and corner detection from scratch. You can use instead the functionality provided by the OpenCV package, namely: `cv2.Canny()` and `cv2.cornerHarris()`.

Experimenting with edge detection. Use your code to figure out appropriate parameter values (or ranges) needed for good performance of edge detection on different imaging scenarios. A good edge detection is one that gives you meaningful contours of objects in the scene, regardless of conditions such as lightning (poor lightning, indoor, outdoor), object-to-camera distance (scale), and blur. A bad edge detection is one that shows every discontinuity as an edge and fills the frame with lots of little wiggly edges.

Experimenting with corner detection. Just as in edge detection, use your code to figure out appropriate parameter values (or ranges) needed for good performance of corner detection on different imaging scenarios. A good corner detection is one where detected corners show characteristics of good keypoints, such as invariance to translation, rotation, and (to some degree) lightning. Choose targets, like grids, etc. that you can manipulate by hand in front of the camera and where you can identify meaningful corners yourself.

2 SIFT Descriptors and Scaling

Programming part. Implement Python code that extracts SIFT keypoints and draws the keypoints back onto the image. You can use either images that you load through `cv2.imread()` or video frames. As before, you don't need to program SIFT from scratch, but please feel free to do so if you want to and let me know so that I can offer guidance.

OpenCV has a couple of implementations of the SIFT algorithm (`cv2.xfeatures2d.SIFT_create()`, and `cv2.ORB_create()`). However, versions of CV2 beyond 3.4.2 no longer offer them because of patent issues. Fortunately, you can install previous versions of OpenCV easily from the terminal within Pycharm. This is what I did in my computer:

```
$ pip install opencv-python==3.4.2.17
$ pip install opencv-contrib-python==3.4.2.17
```

Experimenting with SIFT descriptors Use your code to test the performance of SIFT keypoints relative to translation, rotation, and scale (the most important). You will need to experiment with the parameter values for SIFT in order to get the best performance possible relative to your target images. As with edges and corners, try to figure out the performance limits relative to changes in scale.

3 Keypoints and Matching

Programming part. Implement Python code that uses Harris corners and SIFT keypoints for image matching. Harris corners are keypoints and you can use OpenCV SIFT functions to extract descriptors from these keypoints.

Comparing matching performance. User your code to match two images of the same scene, taken from different angles and distances, using Harris corners as keypoints. Try several examples and try to determine when Harris corners might be useful. Similarly, use your code to match the same images using SIFT keypoints. Their performance for matching should be better and you should confirm that.