C++ for Computer Vision

Using OpenCV - Lecture 7

Problem statement and use case

Have to finish use cases and problem statements today!

- Problem statement:

Good one:

- I want to use contour detection to find flasks.
  - Why correct: You can show contours around a flask as a success scenario

Bad one:

- I want to use feature matching on images if bats
  - Why bad: Not a testable hypothesis.
  - Fix: What is the feature you are looking for?

Problem statements must be 3 unique, but they can use the same images (if they are different problems that are solved)

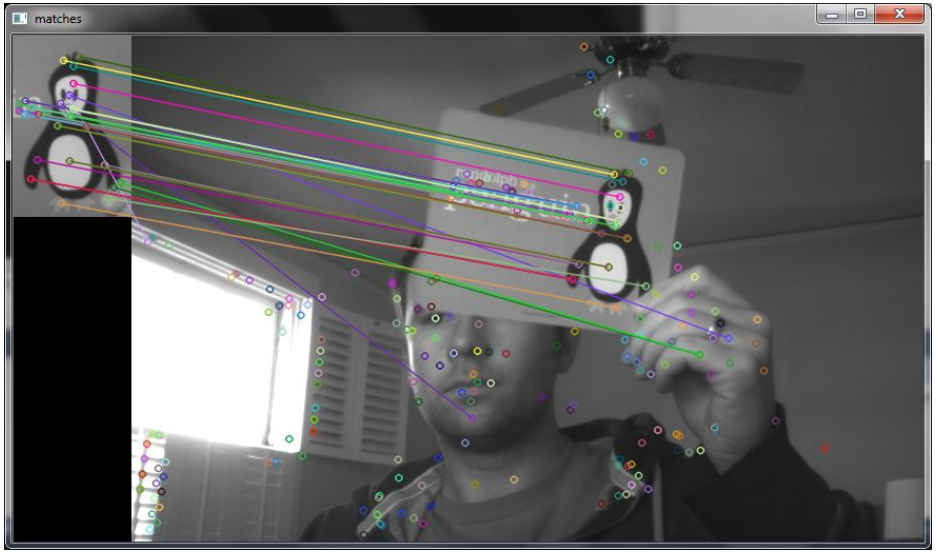
SIFT Features

And keypoint matching

SIFT - Scale-Invariant Feature Transform

Applications include:

- Object recognition,
- Robotic mapping and navigation,
- Image stitching,
- 3D modeling,
- Gesture recognition,
- Video tracking,
- Individual identification of wildlife,
- Etc.



What is SIFT?

- Makes edge maps of several sizes of input image
- Creates matches based on corners and intensity shifts
- Filters out by nearest neighbours and a specialized voting system
- Filters out outliers
- Then returns best matches as keypoints

What to do with all the keypoints?

- There are probably too many matches no matter what you do, and false positives
- You can filter out by kNN (included in code in keypoint_matches.h)
- You can count number of matches in specific area, and only use *x* matches in *y*z* area
- Success criteria:
  - Most matched keypoints are on the object of interest
    - The rest can usually be filtered

Why SIFT?

- Works at different scales
- Widely implemented
- In object matching with little hand-coding, it is unrivalled still
- Great OpenCV support
- Works for surprisingly many applications out of the box, using DescriptorMatcher

Why not SIFT?

- Can be expensive on resources
- By itself, it is useless (keypoints need extra filtering to be useful in most cases)
- Patented (use ORB, SURF or another alternative for industrial applications)
- Black box (not REALLY, but extremely complicated inner workings)
- One-shot - if it does not work directly, and with tweaking with kNN or BF, then give up, use something else

Keypoint matching

- Using brute force (BF) or Flann
- Example in code uses Flann, and is the recommended one for performance
- BF can be used if Flann doesn’t work
- Make sure to resize images if large (SIFT does not scale well with resolution)

Use the function sift_matching() from the keypoint_matching.h file to test. Takes 2 input images - the images where you want to match objects