CSC420 Intro to Image Understanding

Assignment 2 Fall 2018

Posted: October 3, Due October 10

Submit your solutions in a pdf or doc file on MarkUs, along with your code. Include images of your output. You can use built-in functions from OpenCV, Numpy, Scipy, Matlab, etc. unless it is stated that you should write your own. For each piece of code, specify the question to which it corresponds.

1. Interest Point Detection

(a) **[2 points]** Write two functions for computing the Harris corner metric using Harris (R) and Brown (harmonic mean) methods. Display your results for the attached image **building.jpg** showing your cornerness metric output. Compare the results corresponding to two different methods. For Harris you can use the code provided in Tutorial C.

(b) **[2 points]** Write your own function to perform non-maximal suppression using your own functions of choice. Use a circular element, and experiment with varying radii r as a parameter for the output of harmonic mean method. Explain why/how the results change with r. MATLAB users may want to use functions ***ordfilt2()*****,** however it can be easily implemented.

(c) **[2 points]** Write code to search the image for scale-invariant interest point (i.e. blob) detection using the Laplacian of Gaussian and checking a pixel's local neighbourhood as in SIFT. You must find extrema in both location and scale. Find the appropriate parameter settings, and display your keypoints for **synthetic.png** using harmonic mean metric**.** *Hint: Only investigate pixels with LoG above or below a threshold.*

(d) **[1 point]** Use open-source implementation of another local feature descriptor that is not covered in the class, and show the output keypoints on **synthetic.png** and **building.jpg.** Describe the main ideas of your algorithm of choice in a few sentences. You may want to look at **Slide 7** in **Lecture 8-A** for a list of existing methods.

2. SIFT Matching (For this question you will use interest point detection for matching using SIFT. You may use a SIFT implementation (e.g. http://www.vlfeat.org/), or another, but specify what you use)

(a) **[0.5 points]** Extract SIFT keypoints and features for **book.jpg** and **findBook.jpg**.

(b) **[1.5 points]** Write your own matching algorithm to establish feature correspondence between the two images using the reliability ratio on **Lecture 8**. You can use any function for computing the distance, but you must find the matches yourself. Plot the percentage of true matches as a function of threshold. Also, after experimenting with different thresholds, report the best value.

(c) **[2 points]** Use the top k correspondences from part (b) to solve for the affine transformation between the features in the two images via least squares using the Moore-Penrose pseudo inverse. What is the minimum k required for solving the transformation? Demonstrate your results for various k. Use only basic linear algebra libraries.

(d) **[0.5 points]** Visualize the affine transformation. Do this visualization by taking the four corners of the reference image, transforming them via the computed affine transformation to the points in the second image, and plotting those transformed points. Please also plot the edges between the points to indicate the parallelogram. If you are unsure what the instruction is, please look at Figure 12 of [Lowe, 2004].

(e) **[1.5 points]** Write code to perform matching that takes the colour in the images into account during SIFT feature calculation and matching. Explain the rationale behind your approach. Use **colourTemplate.png** and **colourSearch.png**, display your matches with the approach described in part (d).

3. RANSAC

(a) **[0.5 points]** Assuming a fixed percentage of inliers $p = 0.7$, plot the required number of RANSAC iterations (S) to recover the correct model with a higher than 99% chance (P), as a function of k (1:20), the minimum number of sample points used to form a hypothesis.

(b) **[0.5 points]** Assuming a fixed number of sample points required ($k = 5$), plot S as a function of the percentage of inliers p (0.1 : 0.5)

(c) **[1 point]** If $k = 5$ and the initial estimate on the percentage of inliers is $p = 0.2$, what is the the required number of iterations to recover the correct model with $P \geq 0.99$ chance? Assume that you have implemented this and there are 1500 matches in total. In iteration #15, 450 points agree with the current hypothesis (i.e. their error is within a preselected threshold), would the number of required iterations chance? explain how and why.