Intro to Image Understanding (CSC420)

Assignment 2

Posted Sept Oct. 3, 2017; Submission Deadline: Oct. 11th 11.59pm 2017

Instructions for submission: Please write a document and submit a **PDF** with your solutions (include pictures where needed). Include your code inside the document, and submit through **MarkUS**.

For full marks you must show your work, not just your final answer.

Max points: 13, max extra credit points: 3

1. Interest point detection:

   (a) **[2 points]** Write your own function for Harris corner metric using the harmonic mean (slide 30, lecture 6). Display your result for the attached image **building.jpg** showing your cornerness metric output. You can use built-in functions for convolution, gradients, but you must compute M yourself. Adjust α to get a good result.

   (b) **[2 points]** Write your own function to perform non-maximal suppression using **ordfilt2.m** or your own morphological operators function of choice. Use a circular element, and experiment with varying radii r as a parameter. Explain why/how the results change with r.

   (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 may use code from **tutorial 3** as a starting point. You must find extrema in both location and scale. Find the appropriate parameter settings, and display your keypoints for **synthetic.png**. *Hint: Only investigate pixels with the LoG above or below a threshold.*

   (d) **[1 point]** Compare and contrast the Harris corner metric with non-maximal supression as a keypoint detector to the Laplacian of Gaussian method. Show examples where they detect different keypoints and the same keypoints and explain why they are the same/different using **synthetic.png** and **building.png**.

2. 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 **pdist2.m**, but you must find the matches yourself. Experiment for different thresholds.

(c) **[2 points]** Affine transformation: 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 psudeo inverse. Demonstrate your results for various k. Use only basic linear algebra libraries.

(d) **[0.5 point]** 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 rational behind your approach. Use **colourTemplate.png** and **colourSearch.png**, display your matches with (2.d).

3. **Extra: [3 points] total** (this is an optional exercise)

(a) **[1.5 points]** Figure out if Laplacian of Gaussian (LoG) is separable. Do it in two ways.

- **[1 point]** By Brute Force. Write a function which outputs a kernel for LoG (given a σ). Since you do not know the size of the kernel, your function should also take in a tolerance percentage as an input to estimate the effective size of the kernel i.e. if the absolute value of the kernel at a radius drops below a particular percentage of the central value of the kernel, that is where you should stop. Using this approximate kernel, use SVD to figure out if it is separable or not.
- **[0.5 points]** Whatever conclusion you made in part (a), can you arrive at the same conclusion without taking the brute force route i.e by arguing about the expression for the LoG operation.

(b) **[1.5 points]** In the class we saw difference of gaussians (DoG) was a close approximation for LoG. Investigate if this approximation varies as a function of the difference of two sigmas for which DoG was computed. Do this only in 1D.

- **[0.25 points]** Write a function to return a 1D gaussian kernel with a given σ and k such that $x \in [-k, k]$. Use increments of less than 1 (e.g. 0.1) to get a more continuous version.
- **[0.25 points]** Write a function to return a 1D LoG kernel for a given σ, and k.
- **[1 point]** Choose a σ and k of your choice. Compute LoG kernel. Compute a gaussian kernel for σ, call it G_1. Then compute G_2 for a slightly lower σ e.g. if for G_1, σ was 2, set G_2 to a gaussian with $\sigma = 1$. Then compute DoG as

$G_1 - G_2$. Plot LoG and DoG to see if they agree. Vary the difference between the σs, and figure out the difference for which LoG and DoG agree the best. Report the two σs.