CSC420 Intro to Image Understanding Assignment 2 Fall 2019

Post Date: October 1 Due Date: October 8

Instructions for submission: Please write a document in a pdf or doc file with your solutions (include images where needed), and submit through MarkUS. Please include your code and specify the question to which it corresponds.

1. UpSampling

(a) **[0.5 points]** Write your own function to perform one dimensional linear interpolation to quadruple the image size just in one dimension. Apply this function twice in two different dimensions on **bee.jpg** and report the final result. Can we do the same operation with a single 2-dimensional filter? If yes, report the two-dimensional filter if no explain.
You cannot use IMRESIZE in Matlab or resize function in opencv.

(b) [0.5 points] Generalize the one dimensional linear interpolation reconstruction filter designed in lecture 5 to two dimensional case. Justify the designed filter and display your result for the attached image **bee.jpg**. Compare the result with part (a).
You cannot use IMRESIZE in Matlab or resize function in opencv.

2. Interest Point Detection

(a) **[1 points]** Write two functions to detect corners based on Harris (R) and Brown (harmonic mean) metrics. Try to tune the alpha value to achieve the best result on the attached image (**building.jpg)**. Display your results for the attached image and compare the two different metrics. Can you implement these two functions without using determinant and trace? If yes, implement them again and if no, explain why. You can use the presented code in Tutorial B.

(b) [**1 points]** Run your Harris corner detector based on R metric from part (a) on **building.jpg** and 60 degree rotated version of that. Do the corners rotate by the exact same degree? Justify your answer.

(c) **[2 points]** Write a code to search the image for scale-invariant interest points using the Laplacian of Gaussian in different scales as in SIFT. You must find extrema in both position and scale. Display your keypoints for **building.jpg**. *Hint: Only investigate pixels with LoG above or below a threshold.*
You cannot use SIFT package in openCV or detect[a-zA-Z]+Features functions from Matlab.

(d) **[1 points]** In **Slide 6** in **Lecture 8** there is a list of well-known local feature descriptors. Select one of the descriptors that is not covered in the class, and describe the main ideas of the algorithm. State the steps of the algorithm in a bulleted list.

3. Laplacian of Gaussian

(a) **[0.5 points]** Derive the closed-form expression for a 2D Laplacian of Gaussian (LoG) filter. Is LoG a separable filter? Justify your answer.
(b) **[1 point]** Mathematically show how a 2D LoG filter (with std σ) can be approximated by a 2D Difference of Gaussian (DoG) filter (with std $\sigma 1$ and $\sigma 2$). Explain how the choices of $\sigma 1$ and $\sigma 2$ impact the approximation.

4. SIFT Matching [For this question you will use interest point detection for matching using SIFT. You may use any open source SIFT implementation (e.g. OpenCV or VLFeat), but make sure to specify what you used]

(a) **[0.5 points]** Extract SIFT keypoints and features for **sample1.jpg** and **sample2.jpg**. You can transform the provided images to grayscale domain.
(b) **[1 point]** Write your own matching algorithm to establish feature correspondences between the two images using the reliability ratio on **Lecture 8**. Use L2-norm for computing the distance of feature vectors. Plot the number of matches as a function of threshold in a broad range of values. Also, after experimenting with different thresholds, report the best value and visualize the top 10 matches by drawing lines between the correspondences in those two images.
(c) **[1 point]** Repeat part (b) for two different distance functions: L1-norm and L3-norm. Compare the results against Euclidean distance. Which one performs better on the sample images. Intuitively, justify your observations.
(d) **[1 point]** Normalize the given images to be between zero and one, and then generate two noisy images from them by adding zero-mean random gaussian noise with standard deviation 0.08. You also need to clip the output images to make sure all the pixels lie between zero and one. Repeat part (a) and (b) for the new images. Explain how noise affects both the detection and matching phases.
(e) **[1 point]** Write code to perform matching that takes the colour (RGB channels) in the images into account during SIFT feature calculation and matching. Explain the rationale behind your approach. Use **colourTemplate.png** and **colourSearch.png** to test your code; display your matches with the approach described in part (b).