COMP 558 Assignment 2

Prepared by Prof. Siddiqi

Posted: Friday Oct. 25th, 2019
Due: Sunday November 11th, 2019 (midnight)

Introduction

This assignment covers the material from lectures 7 – 13, with lecture 10 excluded.

In order to do the assignment, you need to know how to use Matlab and to be familiar with indexing conventions for matrices, with plots and with the various image processing commands. Use Matlab's documentation for guidance. In particular, typing "doc functioname" will give you detailed information about "functioname" and its variants. If you click on http://mathworks.com, you will also find webinars and tutorials for various computer vision and image processing tasks.

Please use mycourses->Discussions for questions related to this assignment. We expect questions to arise and answers will be posted here. You are also free to discuss the questions with one another. *However, the solutions that you submit must represent your own work.* You are not permitted to copy code from each other, or from the internet. In order to receive full points, your solution code must be properly commented, and you must provide a clear and concise description of your computed results in the PDF file. The TAs will spend at most 20 minutes grading each assignment.

Instructions

Submit a single zipped file **A2.zip** to the mycourses-> Assignment 2 folder. The zip file must contain:

- A PDF with figures and explanatory text and discussion for each question. Whereas you don't have to spend time on elaborate typesetting, please do make sure that your explanations and discussions are clear and that the figures are easy to read and interpret. Also, make sure that all your figures are embedded in your PDF document so that the TAs can provide feedback.
- The Matlab code that you wrote for each part.
- Any images that you used.

Late assignment policy: Late assignments will be accepted up to only 3 days late and will be penalized by 10 percentage points per day late, e.g. An 80 out of 100 would be reduced to 72 for being one day late.

(Pyramids, SIFT Keypoints, SIFT Feature Vectors, Matching) 100 points

For this assignment you will use the "Manor" jpg image as I(x,y), which is a 1024x1024 photograph of the Mystic Manor palace in Hong Kong Disneyland. When you read it into Matlab use only one of its color channels for further processing. Alternatively, you can convert it to greyscale by using the `rgb2gray` function. For all upsampling or downsampling steps you should use Matlab's `imresize` function. For rotations you can use Matlab's `imrotate` function, but be careful because this function carries out rotations about the centre of the image. For generating scaled versions of an image, again use the `imresize` function, but keep in mind that this resizes an image about its centre. The Matlab function `imcrop` may come in handy when you want to create a version of an original image that is the same size but where a scaling and a rotation have occurred about a selected point in the original.

1. [**10 marks**] Generate 7 levels in a Gaussian pyramid, with each scale level given by 2^n, where n is the level, and there is a downsampling factor of 2^n in each direction. In other words, you should first have the original image which is (1024x1024, $\sigma = 0$). Then level 0 is the samesize, i.e., no downsampling, (1024x1024, $\sigma = 1$). Then you have level 1 (512x512, $\sigma = 2$), then level 2 (256x256, $\sigma = 4$) and so on, with the last image being at level 6 (16x16, $\sigma = 64$). Present your results in a composite figure. Qualitatively your results should resemble the examples in the class lecture notes.

2. [**10 marks**] Now, generate 7 levels in a Laplacian pyramid, using the same scale levels as in part 1. For this use the exact process described in the class notes, where in fact each level in the pyramid is generated by a difference of Gaussians at successive scales. To compute this difference, upsampling is required. Once again, present your results in a composite figure. Qualitatively your results should resemble the examples in the class lecture notes.

Explanatory Note: Why is it that the method we described to generate a Laplacian pyramid, *does not actually use convolutions with a Laplacian of Gaussian*? The answer is that there is a close relationship between convolving an image I(x,y) with a difference of Gaussians, $G(x, y, k\sigma) - G(x, y, \sigma)$ (here k is a multiplicative factor), and convolving it with a Laplacian of a Gaussian. The relationship was originally pointed out by Lindeberg in his discussion of scale-normalized derivatives, and was later picked up by David Lowe in his IJCV 2004 paper "Distinctive Image Features from Scale-Invariant Keypoints".

3. [**20 marks**] You will now use the Laplacian pyramid you generated in part 2, to find SIFT keypoint locations. Recall that this is done by searching over all pyramid levels to find pixel locations (x,y) where the intensity I(x,y) has an extremal value (is a local minimum or a maximum) both in space (at that level in the pyramid) and also in scale with respect to the levels above and below (slide 28 in lecture 9). To do the comparisons across scales you need upsampling and downsampling. For finding the extrema you can implement this strategy in a very local sense (experiment with a 3x3 or 5x5 spatial neighborhood). However, you might also want the values at extrema to be distinct from their neighbors in scale space by a *threshold* amount, i.e., to detect only *strong* extrema.

Each keypoint should be represented by a 3-tuple (x, y, σ) and the full set of detected keypoints should be stored in an Nx3 matrix (assuming that you find N keypoints). Illustrate your keypoints by drawing circles at their locations overlayed on the original 1024x1024 image. You will have 5 scales to consider which correspond to keypoints at levels 1,2,3,4 and 5 from part 1. For each scale level use a circle of a different color: blue (level 1), green (level 2), yellow (level 3), magenta (level 4), red (level 5).

4. [**10 marks**] You will now extend your results to compute SIFT feature vectors. You might want to begin by reviewing lecture 9, and in particular slides 29 and onwards. For each SIFT keypoint choose the image in the Gaussian pyramid at its scale. Using a 16x16 window centred at the keypoint, for each pixel in this neighborhood calculate: 1) the gradient magnitude (using central differences), 2) the gradient orientation, 3) a 2D Gaussian weighted version of the gradient magnitude (use a σ of your choice). For illustrative purposes, for a selected keypoint, create a visualization of the 16x16 patch, image gradient magnitude, image gradient orientation and weighted gradient magnitude (as in Lecture 9, slides 31,32).

5. [**15 marks**] Now, based on the results in part 4, create an orientation histogram for each SIFT keypoint, using the weighted gradient magnitudes you have just created. Use 36 orientation bins (corresponding to angles 0,10,...360 degrees). The basic idea is to add up the weighted gradient magnitudes of all gradients in the 16x16 neighborhood whose orientations fall within that particular bin. For the same keypoint you used in part 4, create a plot of the 1D orientation histogram (as in Lecture 9, slide 33). Now use Matlab's `findpeak` function to find the peaks in the orientation histogram for each keypoint. Select the strongest peak and use it as the first entry in the histogram vector, using a CCW direction for the other entries. Essentially this implements a cyclic shift whereby the first entry in the vector corresponds to the local image gradient direction with highest weight. At the end of this part, each SIFT keypoint should now be a SIFT feature vector, represented by a 39-tuple $(\mathrm{x}, \mathrm{y}, \sigma, w0, w1, \dots, w35)$. The additional 36 entries $w0, w1, \dots, w35$ are the entries in the peak-aligned histogram vector.

6. [**10 marks**] Now create two examples of rotated AND scaled versions of the original "Manor" image. Do this by writing a Matlab function that takes five arguments as inputs (image,x0,y0,theta,s), where the second two arguments represent the location about which the transformation is to occur, and s represents the scale factor (greater than 1 for local expansion, less than 1 for local contraction). The function should transform the image by applying the tranformation to it. You are free to use built in Matlab functions for this part, but be careful because some of them carry out operations about the centre of the image while you want the operation to be with respect to your chosen location (x0,y0). Do not worry too much about boundary effects, you can fill in missing information with zeros in the transformed image. Illustrate your results with sample visualizations of the original and the transformed image, clearly describing the amount of rotation and scaling, and marking the location of (x0,y0). For these illustrations and for the next parts of the assignment, pick locations that are somewhat close to the centre of the original image.

7. [**15 marks**] You now have a way of computing SIFT feature vectors for an image. We will now evaluate a somewhat brute force greedy way to carry out SIFT feature matching. For this you will use your original "man" image and the two examples scaled and rotated versions created in part 6. Pick a region of interest in the original image, centred at (x0,y0), the location you used for the image transformation in part 6. For each SIFT feature vector in the original image greedily select the SIFT feature vector in the transformed image, which is its best match in the sense of maximizing the Bhattacharya coefficient between their associated feature vectors. For this part you can restrict your search in the transformed image to an appropriate region in the transformed image. Also, note that the Bhattacharya coefficient is defined for a probability distribution, so the entries in the SIFT feature vector histograms must be normalized so that they add up to 1. Show your results in a creative way, e.g., draw matching lines between "original" SIFT feature vector keypoint locations and their matches in the transformed image. You should interpret your results in a sensible way, e.g., if a SIFT feature doesn't have a good match you could simply treat it as an outlier. Conversely, you could emphasize the good matches over the weaker ones.

8. [**5 marks**] The above SIFT feature vector matching method is carried out in isolation, e.g., a particular match ignores the way in which neighboring SIFT features are matched. Discuss possible ways in which this matching strategy could be made more robust, and the assumptions or additional information you might need to implement such a strategy.