CS485/685 Computer Vision

Spring 2011 – Dr. George Bebis

Programming Assignment 4

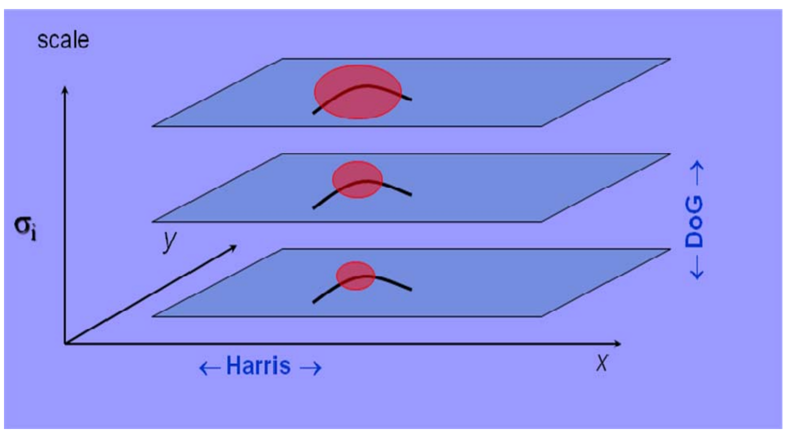
Due Date: 4/5/2011

In this assignment, you will implement the Harris-Laplace interest point detector to extract scale invariant regions.

Algorithm Outline

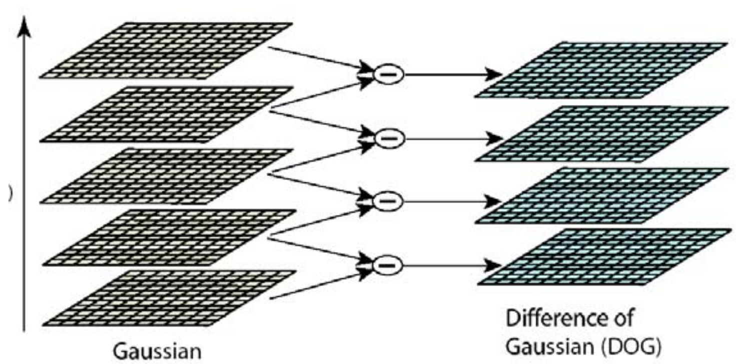
Here is an outline of the main steps; for more details, look at the lecture slides and [1]. You will be using a DoG Scale Space to approximate the Laplacian of Gaussian (LoG) Scale Space.

1. Generate a Gaussian Scale Space (G_SS)
2. Generate a Difference-of-Gaussian Scale Space (DoG_SS)
3. At each level of the G_SS, extract a set of interest points using the Harris corner detector.
4. Perform non-maxima suppression at each scale to find the strongest corners.
5. Perform non-maxima suppression along scales to find most stable corners and extract their characteristic scale.



Steps 1 and 2

You will be using your code from programming assignment 2 to create the G_SS and DoG_SS. To create the G_SS, start from some initial scale σ_0 and iterate n times by increasing the scale by a factor k each time (i.e., $\sigma_0, k\sigma_0, k^2\sigma_0, \ldots, k^n\sigma_0$). You have to choose the initial scale σ_0, the factor k by which the scale is multiplied each time, and the number of levels in the scale space. Set σ_0=2,

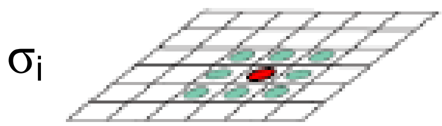


and use 10 to 15 levels in the scale pyramid. The multiplication factor should depend on the largest scale at which you want regions to be detected. If, for example, the largest scale is σ_{MAX}, then $k^n\sigma_0 = \sigma_{MAX}$ or $k^n = \sigma_{MAX}/\sigma_0$ or $k = (\sigma_{MAX}/\sigma_0)^{1/n}$. Once you have built G_SS, DoG_SS can be built by subtracting adjacent levels in G_SS as you did in assignment 2 and shown in the figure above.

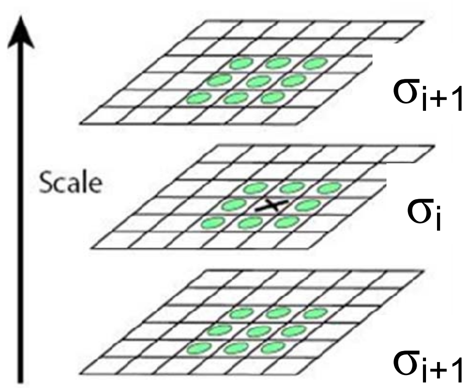
Step 3
Use *cvCornerHarris()* from OpenCV to compute a set of interest points (i.e., corners) at each level of the G_SS. There are several parameters to set including the integration scale σ_I (i.e., set neighborhood size $5\sigma_I$), the differentiation scale σ_D (i.e., set mask size size $5\sigma_D$), and α (i.e., see below; use the default value which is 0.04). Typically, if $\sigma_i = k^i\sigma_0$ at the i-th level of the G-SS, then $\sigma_I = \sigma_i$ and $\sigma_D = 0.7\sigma_i$. OpenCV uses the same formula we discussed in class to determine cornerness:

$$\mathbf{R(A_W) = det(A_W) - \alpha\, trace^2(A_W)}$$

Step 4
For each interest point detected in step 3, determine whether it corresponds to a local maximum by comparing its $\mathbf{R(A_W)}$ value to the $\mathbf{R(A_W)}$ values of its eight neighbors (e.g., use a 3 x 3 window) as shown in the figure below. Reject corners which do not correspond to local maxima **or** their $\mathbf{R(A_W)}$ value is less than a threshold $\mathbf{t_1}$.



Step 5
For each interest point detected in step 3, determine whether it corresponds to a local maximum by comparing its DoG value to the DoG values of its eight neighbors at the same scale as well as the nine neighbors at the scale above and the nine neighbors at the scale below (i.e., 26 neighbors) as shown in the figure below. Reject corners which do not correspond to local maxima **or** their DoG value is less than a threshold $\mathbf{t_2}$.



Displaying interest points

You need to display each interest point along with its spatial extent (i.e., characteristic scale) using a circle of radius r where σ corresponds at the scale (level) from which the interest point was detected. For clarity, show the location of each interest point using a distinctive symbol (e.g., cross, small rectangle). The interest points should be shown on the original image. You can use *cvCircle()* for drawing the circles around the interest points.

$$r = \sqrt{2}\sigma$$

Experiments

Use the images provided on the course's webpage. You need to experiment with different values for t_1 and t_2 and report your best results. For each image, display the interest points after (i) step 3, (ii) step 4, and (iii) matching interest points only. In each case, report the number of interest points before and after applying the thresholds and non-maxima suppression steps. To find which corners match between images, use the transformation matrix provided (i.e., homography) for different pairs of images to map the corners from one image onto the other. Two points $\mathbf{x}_a$ and $\mathbf{x}_b$ correspond if the error in relative point location is less than 1.5 pixel: $||\mathbf{x}_a - H\,\mathbf{x}_b|| < 1.5$, where H is the homography between the images.

Graduate Students Only

Implement the affine adaptation step to turn circular blobs into ellipses as shown in the lecture (just one iteration is sufficient). Note that each region should be adapted separately. The selection of the correct window function is essential here. You should use a Gaussian window that is a factor of 1.5 or 2 larger than the characteristic scale of the blob. Note that the lecture slides show how to find the relative shape of the second moment ellipse, but not the absolute scale (i.e., the axis lengths are defined up to some arbitrary constant multiplier). A good choice for the absolute scale is to set the sum of the major and minor axis half-lengths to the diameter of the corresponding DoG circle. To display the resulting ellipses, use *cvEllipse().* Show your results before and after the affine iterations.

What to turn in

You are to turn in a report including a print-out of your source code. Your report should include the following: a description of the experiments, results (i.e., include graphic output of your results), discussion of results and comparisons, and a brief summary of what you have learned. **The report is very important in determining your grade for the programming assignment.** Be well organized, type your reports, and include figure captions with a brief description for all the figures included in your report. Motivation and initiative are greatly encouraged and will earn extra points.

References

[1] K. Mikolajczyk and C. Schmid, "Scale and Affine Invariant Interest Point Detectors", *International Journal of Computer Vision*, 60(1), pp. 63-86, 2004.