MVA 2008 - 2009

Object recognition and computer vision

Assignment 1: Scale-invariant blob detection

Jean Ponce, Cordelia Schmid and Josef Sivic

(adapted from Svetlana Lazebnik, UNC)

Due date: October 21st 2008

The goal of the assignment is to implement a Laplacian blob detector as discussed in the lectures.

Algorithm outline:

1. Generate a Laplacian of Gaussian filter.
2. Build a Laplacian scale space, starting with some initial scale and going for n iterations:
    1. Filter image with scale-normalized Laplacian at current scale.
    2. Save square of Laplacian response for current level of scale space.
    3. Increase scale by a factor k.
3. Perform non-maximum suppression in scale space. **(See the note below)**.
4. Display resulting circles at their characteristic scales.

Tips

· It is relatively inefficient to repeatedly filter the image with a kernel of increasing size. Instead of increasing the kernel size by a factor of k, try downsampling the image by a factor 1/k (of course, then you will have to upsample the result or do some interpolation in order to find maxima in scale space). Alternatively, implement the difference-of-Gaussian pyramid as described in section 3 of David Lowe's paper (see below). Remember that

· NEW: **On the non-maximum suppression in scale space.** The goal is to find pixels which are local maxima in the scale-space. This amounts to finding pixels with the filter response (strictly) greater than its 26 scale-space neighbours, considering also the adjacent scales as illustrated in figure 2 of David Lowe's paper. When the scale-space is obtained by filtering the image with a kernel of increasing size, as described in the algorithm outline above, all levels of the scale-space have the same image size and you only need to check the 3x3 spatial neighbourhood at the same pixel location in the adjacent levels. When the scale-space is obtained by image down-sampling while keeping the filter size constant, you need to locate the correct position of the spatial neighbours in the adjacent levels. This can be achieved by up-sampling / down-sampling the adjacent levels to the same size as the current level of the pyramid, effectively doing interpolation. Alternatively, you can just consider the nearest 3x3 spatial neighbourhood in the adjacent levels. This avoids the (potentially expensive) interpolation, but is only an approximation.

· You have to choose the initial scale, the factor k by which the scale is multiplied each time, and the number of levels in the scale space. Reasonable value for the initial scale is 2, using 10 to 15 levels for the scale pyramid. The multiplication factor should depend on the largest scale at which you want regions to be detected (or, if you are implementing the difference-of-Gaussian scale space, this factor depends on computational efficiency considerations as described in David Lowe's paper).

· You may want to use a three-dimensional array to represent your scale space. It would be declared as follows:

```
scale_space = zeros(h,w,n); % [h,w] - dimensions of image, n - number of
levels in scale space
```

Then `scale_space(:,:,i)` would give you the i-th level of the scale space. Alternatively, if you are storing different levels of the scale pyramid at different resolutions, you may want to use a cell array, where each "slot" can accommodate a different data type or a matrix of different dimensions. Here is how you would use it:

```
        scale_space    = cell(n,1); %creates a cell array with n "slots"
        scale_space{i} = my_matrix; % store a matrix at level i
```

· To perform nonmaximum suppression in scale space, you may first want to do nonmaximum suppression in each 2D slice separately. For this, you may find functions `nlfilter`, `colfilt` or `ordfilt2` useful. To extract the final nonzero values (corresponding to detected regions), you may want to use the `find` function.

· You also have to set a threshold on the squared Laplacian response above which to report region detections. A reasonable value is 0.001, but you should play around with different values and choose one you like best.

· To display the detected regions as circles, you can use this function (or feel free to write your own). Don't forget that there is a multiplication factor that relates the scale at which a region is detected to the radius of the circle that most closely "approximates" the region.

· If you decide to experiment with different implementation choices for building the scale space and nonmaximum suppression, you may want to compare these choices according to their computational efficiency. To time different routines, use `tic` and `toc` commands.

· Apply the detector to greyscale versions of colour images. In Matlab, you can convert a color image to greyscale using

```
        Im_gray = mean(Im_rgb,3);
```

Test images

Here are four images to test your code. Also run your code on two images of your own choice.

What to hand in

· You should implement the basic version of the Laplacian blob detector as described in the algorithm outline above. Extra credit will be given for implementing one of the more efficient options: (1) using image down-sampling by factor *1/k* (rather then increasing the kernel size by a factor of *k*) or (2) the difference of Gaussian approximation described in section 3 of David Lowe's paper.

You should prepare a (very brief) report including the following:

· An explanation of any "interesting" implementation choices that you made.
· An explanation of parameter values you have tried and which ones you found to work well.
· Output of your detector on all test images for your "favourite" parameter settings.

Send your report, code, and your two additional test images in a single zip file to **Josef Sivic <Josef.Sivic@ens.fr>**.

Helpful resources

· MATLAB tutorials (from Martial Hebert at CMU): basic operations, programming, working with images
· Sample Harris detector code
· Nice Slides by Svetlana Lazebnik on feature detection describing also scale invariant blob detection (slides 32—49).
· Blob detection on Wikipedia.
· D. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60 (2), pp. 91-110, 2004. This paper contains details about efficient implementation of a Difference-of-Gaussians scale

space.

- T. Lindeberg, "Feature detection with automatic scale selection," International Journal of Computer Vision 30 (2), pp. 77-116, 1998. This is somewhat advanced reading for those of you who are *really* interested in the gory mathematical details.