MVA – Object recognition and artificial vision

Assignment 1

Scale Invariant Blob Detection

Jean-Baptiste Fiot

Oct 2008

1 Goal

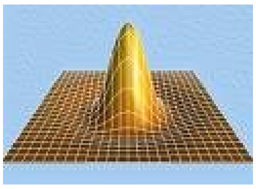
The aim of this project is to develop a scale-invariant blob detector, meaning an algorithm detecting blobs independently of their sizes.

2 How to detect a blob ?

In a picture, a blob is detected by an extrema in the convolution of a Laplacian of Gaussian (LoG) by this image. When we find such an extrema, we have this simple relation :

$$\boxed{r_{blob}=\sqrt{2}*\sigma}$$

2.1 LoG kernel implementation :



► I have implemented two functions :

- LoG.m, which returns the value of laplacian of gaussian at x, y, σ
- 2DLoG.m, which returns a laplacian of gaussian kernel, given the scale σ and the desired size for the kernel. This function relies on LoG.m.

To convolve to get one level of the scale-space, I simply use the conv2 matlab function.

3 First method: several σ by increasing the LoG kernel size

A 1st method is to compute the convolution of the image with laplacians of gaussians with several scales, to find blobs independently of the size of the blob. We store all these convolutions is a space called gaussian scale space.

In my 1st implementation, I have chosen to convolve the level i with a LoG with σ=i. Thus:

$$\boxed{\{Blob\,radius\,detectable\}=\{\sqrt{(2)},2\sqrt{(2)},\ldots,n\sqrt{(2)}\}}$$

Besides, when we increase the scale of the LoG, we have to increase the kernel size (to avoid having just the top part of the Mexican hat in the kernel).

Technically, I have used a ratio $\frac{\text{kernel size}}{\text{LoG scale}}=4$

3.1 Non maximum suppression.

► My 1st method was to eliminate non-maximum positions by checking the 8-neighbors on 2D slices. This basically leads to h*w*n calls to **nlfilter**, giving the c8_non_local as for the function parameter. (I have implemented this function).

This technique was quite slow.

And also, it does not check the 26 neighbors (the 8 corners are missed).

► By using **colfilt** instead of nlfilt, the computing time has decreased by a factor 3.

Another interesting point to notice is that the demanding operations are not on the same slices as the ones with nlfilt. Here the demanding operations are on the scale-slices, which will be reduced with the down-sampling technique.

► I also implemented my own method to do this non-maximum suppression. Instead of doing it by slices, and a sliding window, I implemented c26_non_max.m and vblock.m to do it with a **sliding 3*3*3 block in the scale space**.

- vblock returns a 3*3*3 block centered on a position (i,j,k), zero-padded if need be.
- c26_non_max compares the central value of a 3*3*3 block with his 26 neighbors

This method is the fastest for this step, without missing non maximum suppression points.

3.2 Threshold and displaying the blobs.

Finally, we find the scale_space positions with a value above a chosen threshold, and we display the results with the given function show_all_circles.m.

4 Second Method: Down-sampling.

To decrease computing time, especially during the non-maximum suppression step, we can down-sample the image and convolve with a kernel of reduced scale, thus a reduced size.

Here two possibilities:

- getting each level from the previous one, by a factor 1/k. That gives us:

$$\boxed{\{Blob\, radius\, detectable\} = \{k\sqrt{(2)}, k^2\sqrt{(2)}, \ldots, k^n\sqrt{(2)}\}}$$

- getting the level i by down-sampling (with mean) the first level by a factor 1/i. That gives us:

$$\boxed{\{Blob\, radius\, detectable\} = \{\sqrt{(2)}, 2\sqrt{(2)}, \ldots, n\sqrt{(2)}\}}$$

  The second one is a little bit slower, but provides a better search.

I have implemented my down-sampling function: SubSampling.m, and tried both possibilities: see SIBD2.m and SIBD2_bis.m

4.1 Limitation of the number of scale space levels.

The scale-space level number is limited so that we are able to do the interpolation backwards.

We have:

- for the 1st possibility:

$$min(floor(\frac{h}{k^{n_{max}}}), floor(\frac{w}{k^{n_{max}}}))=2 \Leftrightarrow n_{max}=min(floor(\frac{\log(\frac{h}{2})}{\log(k)}), floor(\frac{\log(\frac{w}{2})}{\log(k)}))$$

- for the 2nd possibility:

$$min(floor(\frac{h}{n_{max}}), floor(\frac{w}{n_{max}}))=2 \Leftrightarrow n_{max}=min(floor(\frac{h}{2}), floor(\frac{w}{2}))$$

This explains the update of n in SIBD2.m

(It is not used in SBD2_bis.m, because given the usual images' sizes and a moderate choice for n, the condition is not critical.)

4.2 Non maximum suppression

Here is what I do:

- non maximum suppression on the upper left parts of the scale-space 2D slices (/3rd coord ie scale).
- Interpolation of these slices back to h*w, using interp2.
- Sliding 3*3*3 block technique to remove all the remaining non-maximum positions.

5 Comparative results

We note the methods:

M1a: convolution with kernel of an increasing size + nlfilt non max suppression

M1b: convolution with kernel of an increasing size + colfilt non max suppression

M1c: convolution with kernel of an increasing size + sliding 3*3*3 block non max suppression

M2a: down-sampling by a factor k from one level to the next one

M2b: down-sampling by a factor i from the 1st image to get the i-th level

5.1 Computing time

I got these times with the butterfly pic, with n=10 scale space levels. (This level is automatically updated to 7 in the M2a method).

	M1a	**M1b**	**M1c**	**M2a**	**M2b**
Scale Space creation	2.9	2.9	2.9	0.6	3.5

Non Maximum Suppression	30.8+112.9+164.8	88.6+6.1+6.1	65.2	2.2+15.0	9.5+13.5
Interpolation	0	0	0	2.2	1.6
Extrema	0.1	0.1	0.1	0.1	0.1
Display	0.4	0.3	0.3	0.4	0.4
TOTAL	**311.4**	**104.1**	**68.5**	**19.8**	**28.6**

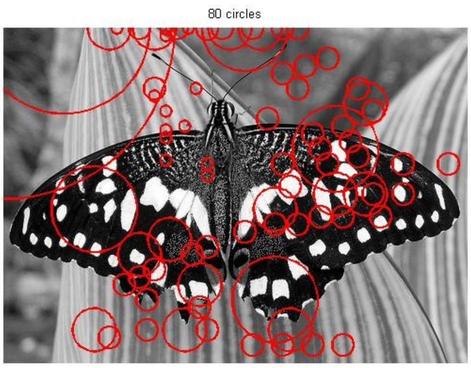
5.2 Blob detection flaws.

M1x methods only detect well really white blobs.

Example:

M2a has a detection range in k^i, and not in i, thus does not detect the blobs of « intermediate » size (between a k^i and k^{i+1}), and does not really give a good visual impression.

Example:



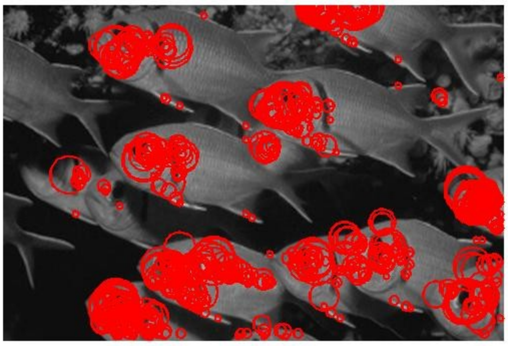
6 M2b method Gallery

The M2b method has proved to give the best results, so here are a few screenshots.

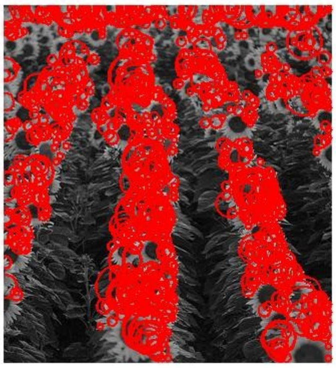
1223 circles

977 circles

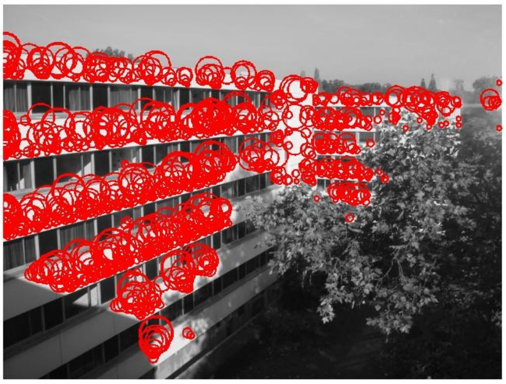
668 circles



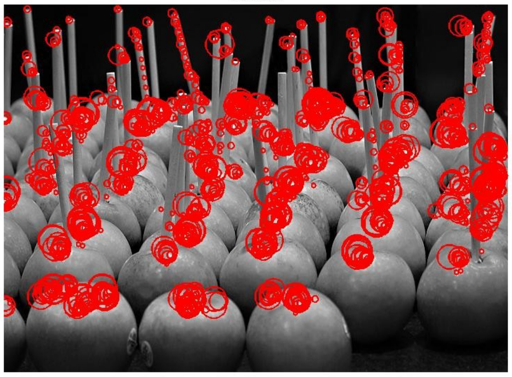
941 circles



907 circles



1332 circles



7 Further studies

- threshold sensibility...
- Difference of Gaussian method