1 Motivation

2 Theory

see also

inf.ed.ac

wikipedia

Java implementation used in trackmate

python scikit-image implementation

The Laplacian of Gaussian filter can be thought of as first applying a Gaussian filter to smooth the image, then taking the Laplacian (spatial second derivative). With the correct choice of Gaussian width σ as $\sigma = r/\sqrt{n}$ in n dimensions ($\sigma = r\sqrt{2}$ in two dimensions), the series of operations transforms the centers of spheres of radius r into distinct intensity maxima/minima. We then identify regional maxima on the filtered image as the locations of blobs in the original image.

The image must be convolved first with the Gaussian kernel, then with the Laplacian operator. Equivalently, the two kernels are first combined to the LoG kernel

$$LG^{2D} = \frac{1}{\pi\sigma^4}\left(-1 + \frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1}$$

in two dimensions,

$$LG^{3D} = \frac{1}{\sqrt{2\pi}^3\sigma^5}\left(-3 + \frac{x^2+y^2+z^2}{\sigma^2}\right)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \tag{2}$$

in three dimensions, and generally

$$LG = \frac{1}{\sqrt{2\pi}^n\sigma^{n+2}}\left(-n + \frac{\sum x_i x^i}{\sigma^2}\right)e^{-\frac{\sum x^i x_i}{2\sigma^2}} \tag{3}$$

in n dimensions, which is then convolved with the image in one step.

To implement the LoG filter in matlab, we construct the kernel as a small 2D or 3D matrix, deriving σ from the user-input estimated diameter/radius of the blobs and filling a matrix with numerical values of LG^{2D} or LG^{3D} at distance (x, y) or (x, y, z) from the center of the kernel, and convolve the kernel with the image using matlab builtin `conv2` or `convn`. The efficient convolution operation probably consists of Fourier transformin image and kernel and multiplying pointwise in Fourier space (convolution theorem).

3 Detailed implementation

First a median filter is applied to the image; this can be disabled by passing name-value argument `MedianFilter=false`.

The LoG kernel is constructed as described above in function `LoG_kernel` or `Log_kernel_3D`. It's an array of dimensions `kernelSize` $\times$ `kernelSize` or `kernelSize` $\times$ `kernelSize` $\times$ `kernelSize` , where kernelsize is chosen as $3 + 2\max(2, \lceil 3\sigma + .05\rceil)$, roughly 6σ. This kernel size includes crucial features of the kernel, the central minimum and the circular maximum; smaller kernels do not effectively apply a LoG filter and lead to worse results.

The image is convolved with the kernel using matlab's builtin `conv2` / `convn` function. Convolution mode `'same'`, which pads borders with zeros to return a image with the same dimensions, is chosen. The convolution is the most costly step for large or 3D images. The matlab default convolution algorithm seems to use something like "overlap-and-save" method, appropriate for cases where the kernel is much smaller than the image. An alternative convolution implementation by fourier transforming both image and kernel and multiplying pointwise in Fourier space (convolution theorem) was found to be slower.

Next the coordinates of prominent maxima in the convolved image must be identified. Following scikit-image, I find regional maxima by applying a maximum filter (`imdilate`) and identifying pixels where the original image is equal to the dilated image.

Particles within one radius of the image edges are deleted, unless a different distance is set as `BorderWidth`. Set as `0` to disable.

Intensity of the central coordinate in the original image of the putative particle is used as a "quality" score. By default particles with 10% of the brightest particle or less are removed. In our cases with grainy images, a cutoff of 30% seems more appropriate. Set as `QualityFilter=0.3`.

Overlapping particle pairs are found and the lower-intensity particle is removed when `OverlapFilter=true` is given. Warning: $O(N^2)$ runtime in number of particles. Can be slow in large or 3D images if the quality filter is not used and large numbers of spurious particles are found.

The result is returned as a $3 \times N$ or $4 \times N$ array with columns (x, y, quality) or (x,y,z,quality).

4 Further

Write a wrapper unifying `blobdetect` and `blobdetect3D`, detecting whether the input is 2D or 3D.

Can some way to accelerate the convolution step be found, especially for 3D images that may be too large for GPU RAM?

Scale-free LoG filter could be implemented to identify spheres of heterogeneous / unknown diameter and their diameters.

Improve overlap detection using k-dimensional tree or information from adjacent time points.