Laplacian/Laplacian of Gaussian

Common Names: Laplacian, Laplacian of Gaussian, LoG, Marr Filter

Brief Description

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection (see zero crossing edge detectors). The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian smoothing filter in order to reduce its sensitivity to noise, and hence the two variants will be described together here. The operator normally takes a single graylevel image as input and produces another graylevel image as output.

How It Works

The Laplacian *L(x,y)* of an image with pixel intensity values *I(x,y)* is given by:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

This can be calculated using a convolution filter.

Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian. Two commonly used small kernels are shown in Figure 1.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Figure 1 Two commonly used discrete approximations to the Laplacian filter. (Note, we have defined the Laplacian using a negative peak because this is more common; however, it is equally valid to use the opposite sign convention.)

Using one of these kernels, the Laplacian can be calculated using standard convolution methods.

Because these kernels are approximating a second derivative measurement on the image, they are very sensitive to noise. To counter this, the image is often Gaussian smoothed before applying the Laplacian filter. This pre-processing step reduces the high frequency noise components prior to the differentiation step.

In fact, since the convolution operation is associative, we can convolve the Gaussian smoothing filter with the Laplacian filter first of all, and then convolve this hybrid filter with the image to achieve the required result. Doing things this way has two advantages:

- Since both the Gaussian and the Laplacian kernels are usually much smaller than the image, this method usually requires far fewer arithmetic operations.
- The LoG (`Laplacian of Gaussian') kernel can be precalculated in advance so only one convolution needs to be performed at run-time on the image.

The 2-D LoG function centered on zero and with Gaussian standard deviation σ has the form:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

and is shown in Figure 2.

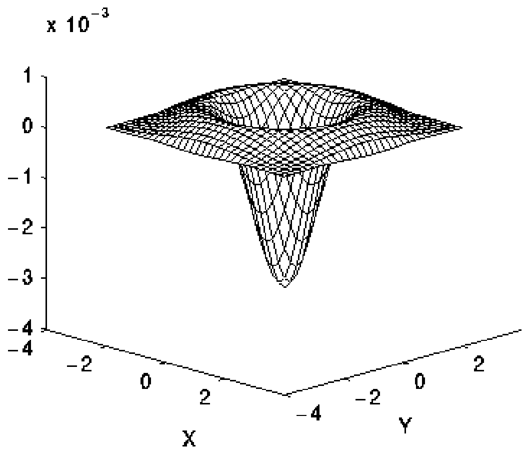
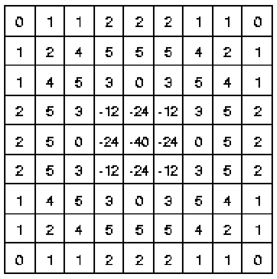


Figure 2 The 2-D Laplacian of Gaussian (LoG) function. The *x* and *y* axes are marked in standard deviations (σ).

A discrete kernel that approximates this function (for a Gaussian σ = 1.4) is shown in Figure 3.



0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Figure 3 Discrete approximation to LoG function with Gaussian σ = 1.4

Note that as the Gaussian is made increasingly narrow, the LoG kernel becomes the same as the simple Laplacian kernels shown in Figure 1. This is because smoothing with a very narrow Gaussian (σ < 0.5 pixels) on a discrete grid has no effect. Hence on a discrete grid, the simple Laplacian can be seen as a limiting case of the LoG for narrow Gaussians.

Guidelines for Use

The LoG operator calculates the second spatial derivative of an image. This means that in areas where the image has a constant intensity (*i.e.* where the intensity gradient is zero), the LoG response will be zero. In the vicinity of a change in intensity, however, the LoG response will be positive on the darker side, and negative on the lighter side. This means that at a reasonably sharp edge between two regions of uniform but different intensities, the LoG response will be:

- zero at a long distance from the edge,
- positive just to one side of the edge,
- negative just to the other side of the edge,
- zero at some point in between, on the edge itself.

Figure 4 illustrates the response of the LoG to a step edge.

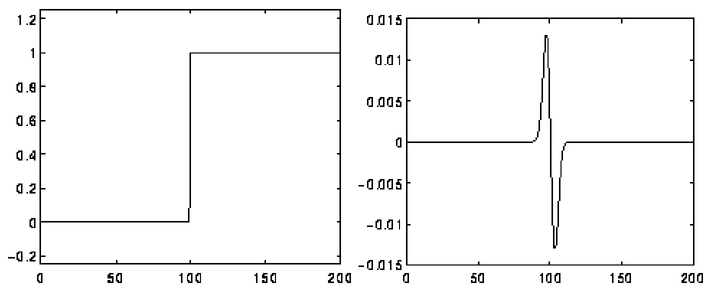
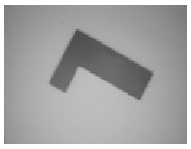


Figure 4 Response of 1-D LoG filter to a step edge. The left hand graph shows a 1-D image, 200 pixels long, containing a step edge. The right hand graph shows the response of a 1-D LoG filter with Gaussian σ = 3 pixels.

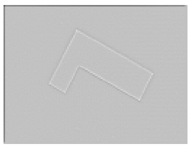
By itself, the effect of the filter is to highlight edges in an image.

For example,



is a simple image with strong edges.

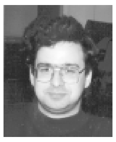
The image



is the result of applying a LoG filter with Gaussian $\sigma = 1.0$. A 7×7 kernel was used. Note that the output contains negative and non-integer values, so for display purposes the image has been normalized to the range 0 - 255.

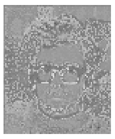
If a portion of the filtered, or gradient, image is added to the original image, then the result will be to make any edges in the original image much sharper and give them more contrast. This is commonly used as an enhancement technique in remote sensing applications.

To see this we start with



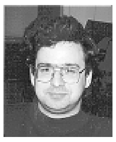
which is a slightly blurry image of a face.

The image



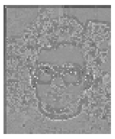
is the effect of applying an LoG filter with Gaussian $\sigma = 1.0$, again using a 7×7 kernel.

Finally,

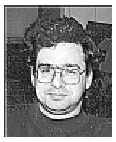


is the result of combining (*i.e.* subtracting) the filtered image and the original image. Note that the filtered image had to be suitable scaled before combining in order to produce a sensible enhancement. Also, it may be necessary to translate the filtered image by half the width of the convolution kernel in both the *x* and *y* directions in order to register the images correctly.

The enhancement has made edges sharper but has also increased the effect of noise. If we simply filter the image with a Laplacian (*i.e.* use a LoG filter with a very narrow Gaussian) we obtain



Performing edge enhancement using this sharpening image yields the noisy result



(Note that unsharp filtering may produce an equivalent result since it can be defined by adding the negative Laplacian image (or any suitable edge image) onto the original.) Conversely, widening the Gaussian smoothing component of the operator can reduce some of this noise, but, at the same time, the enhancement effect becomes less pronounced.

The fact that the output of the filter passes through zero at edges can be used to detect those edges. See the section on zero crossing edge detection.

Note that since the LoG is an isotropic filter, it is not possible to directly extract edge orientation information from the LoG output in the same way that it is for other edge detectors such as the Roberts Cross and Sobel operators.

Convolving with a kernel such as the one shown in Figure 3 can very easily produce output pixel values that are much larger than any of the input pixels values, and which may be negative. Therefore it is important to use an image type (*e.g.* floating point) that supports negative numbers and a large range in order to avoid overflow or saturation. The kernel can also be scaled down by a constant factor in order to reduce the range of output values.

Common Variants

It is possible to approximate the LoG filter with a filter that is just the difference of two differently sized Gaussians. Such a filter is known as a *DoG* filter (short for `Difference of Gaussians').

As an aside it has been suggested (Marr 1982) that LoG filters (actually DoG filters) are important in biological visual processing.

An even cruder approximation to the LoG (but much faster to compute) is the DoB filter (`Difference of Boxes'). This is simply the difference between two mean filters of different sizes. It produces a kind of squared-off approximate version of the LoG.

Interactive Experimentation

You can interactively experiment with the Laplacian operator by clicking here.

You can interactively experiment with the Laplacian of Gaussian operator by clicking here.

Exercises

1. Try the effect of LoG filters using different width Gaussians on the image

   What is the general effect of increasing the Gaussian width? Notice particularly the effect on features of different sizes and thicknesses.

2. Construct a LoG filter where the kernel size is much too small for the chosen Gaussian width (*i.e.* the LoG becomes truncated). What is the effect on the output? In particular what do you notice about the LoG output in different regions each of uniform but different intensities?

3. Devise a rule to determine how big an LoG kernel should be made in relation to the σ of the underlying Gaussian if severe truncation is to be avoided.

4. If you were asked to construct an edge detector that simply looked for peaks (both positive and negative) in the output from an LoG filter, what would such a detector produce as output from a single step edge?

References

R. Haralick and L. Shapiro *Computer and Robot Vision*, Vol. 1, Addison-Wesley Publishing Company, 1992, pp 346 - 351.

B. Horn *Robot Vision*, MIT Press, 1986, Chap. 8.

D. Marr *Vision*, Freeman, 1982, Chap. 2, pp 54 - 78.

D. Vernon *Machine Vision*, Prentice-Hall, 1991, pp 98 - 99, 214.

Local Information

Specific information about this operator may be found here.

More general advice about the local HIPR installation is available in the *Local Information* introductory section.