Edge Detection

Charles de Freitas

Msc Computer Science

School of Computer Science

13th December 2018

Keywords: Filtering, Smoothing

1 LAPLACIAN OF GAUSSIAN

Laplacian of Gaussian refers to convolution of a Gaussian smoothing mask with a Laplacian filter. The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image.[1] A small sample Laplacian as as follows:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

This is an example of a negative peek Laplacian, but inverting all the elements would work too. The change in sign across the Laplacian highlights rapid intensity change due to having such a sharp gradient across itself. Without smoothing, image noise will cause these rapid changes in intensity. Since MATLAB includes a function for calculating the discrete Laplacian, we can simply perform del2(img) on the image, then convolve this onto the Gaussian. (This is because it requires less computations to convolve the two smaller matrices first than to apply them sequentially to the image.) The final step is to refine the edges through zero crossing, this has been achieved by *checking neighbours* of an element to change changes in sign.

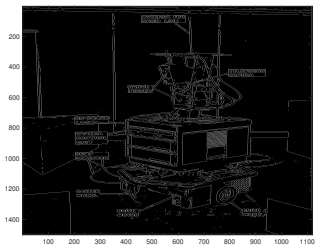
1.1 Laplacian of Gaussian

```
    gaus1d = 1/(s * sqrt(2* pi)) * exp(-(x-m).^2/(2*s^2));
    gaus2d = gaus1d' * gaus1d;
    laplacian = del2(image);
    LoG = conv2(gaus2d, laplacian, 'same');
    LoG_image = conv2(image, LoG, 'same');
    output = zero_cross(LoG_image, 1);
```

1.2 Zero Crossing

```
function output = check_neighbours(src, i, j, t)
    output = 0;
    x = i + 1;
    y = j + 1;
    img = padarray(src,[1 1]);
    if img(x - 1, y) * img(x + 1, y) < 0

        if abs(img(x - 1, y)) + abs(img(x + 1, y)) > t
            output = 1;
        end
    elseif img(x - 1, y - 1) * img(x + 1, y + 1) < 0
        if abs(img(x - 1, y - 1)) + abs(img(x + 1, y + 1)) > t
            output = 1;
        end
    elseif img(x + 1, y + 1) * img(x - 1, y - 1) < 0
        if img(x + 1, y + 1) + img(x - 1, y - 1) > t
            output = 1;
        end
    elseif img(x, y - 1) * img(x, y + 1) < 0
        if abs(img(x, y + 1)) + abs(img(x, y + 1)) > t
            output = 1;
        end
    end
end
```



Changes in sign will represent an edge since the result of **LoG** is a differential image.

The resulting image is Laplacian of Gaussian where the Gaussian is of

$$\bar{x} = 0, \sigma = 2.5$$

This has produced a very clean image, if you wanted

2 CELL DETECTION

2.1 Filters

2.1.1 *Sobel in X*

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

2.1.2 *Sobel in Y*

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

2.1.3 *Roberts in X*

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

2.1.4 *Roberts in Y*

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

2.1.5 *First Order Laplacian*

$$\begin{bmatrix} 0.1897 & 0.1741 & 0 & -0.1741 & -0.1897 \end{bmatrix}$$

ab_max applies the given filter to the image and returns the larges value in the resulting array. This is as to create an appropriate range of thresholds for the image. With this range of thresholds, an array of resultant images is produced.

```
function o = filter(f,img, g)
    space = linspace(0, ab_max(img, g));
    a = arrayfun(@(t) f(img, g, t), space, 'Uniform', 0);
    o = cell2struct(a, 'img');
end

function o = ab_max(img, g)
    smoothed = conv2(img, g, 'same');
    o = max(max(smoothed));
end
```

In the case of Roberts, the end function is:

```
function o = roberts(img, gaus, t)
    load resources robertsX robertsY;
    smoothed = conv2(img, gaus, 'same');
    x = conv2(smoothed, robertsX, 'same');
    y = conv2(smoothed, robertsY, 'same');
    hyp = hypot(x, y);
    o = hyp > t;
    clear robertsX robertsY;
end
```

Whereas for Laplacian you only need to convolve the single 2D matrix onto your smoothed image:

```
function o = laplacian(img, g, t)
    load resources lap;
    smoothed = conv2(img, g, 'same');
    o = abs(conv2(smoothed, lap, 'same')) > t;
end
```

2.2 Variable Smoothing

1. Create a set of Gaussian masks is produced for each filter.
2. Create Ω, a linear space bounded by 0 and the largest value possible for a given smoothed image.
3. Apply the given smoothed filter for ω where $\omega \in \Omega$
4. From the resulting set, calculate **TPR** and **FPR**.
5. Repeate for all Gaussians and return a set of coortinates.
6. Plot the set that contains the shortest distance to $(0, 1)$
7. Repeate for all filters and respective Gaussians.

Tuning Since each filter behaves differently, an appropriate set of Gaussian masks must be selected for each. Starting with fixed values of $\bar{X} = 0, \sigma = 1$ and a range of 0:1:10[2].
Figure 1

3 RESULTS

	Sobel	Roberts	Gaussian	LoG	Laplacian
Specificity	0.8376	0.817	0.8352	0.8164	0.7449
Sensitivity	0.8913	0.9027	0.8814	0.9464	0.5910

Table 1. Scores for '9343 AM.bmp

Sobel, Roberts and First order Gaussian all give very similar results. This is due to them all being differential operators and in effect calculating edges through the same method, with varying degrees of success.

[2] From 0 to 10 and stepping by 1

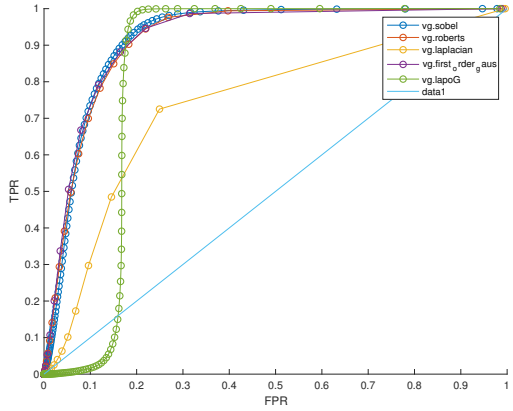


Figure 1. Optimal curves when tested with identical Gaussians of size 0-10 on '9343 AM.bmp'

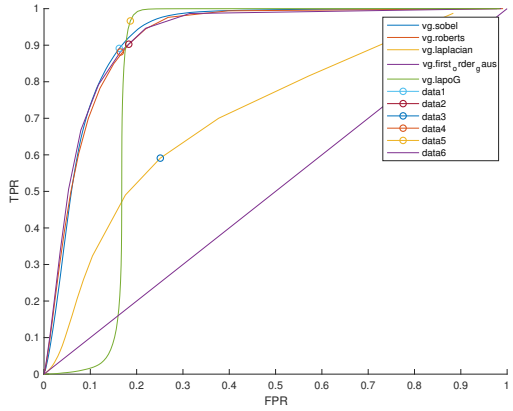


Figure 2. ROC space of each curve with most optimal value highlighted for '9343 AM.bmp'

	Sobel	Roberts	Gaussian	LoG	Laplacian
Specificity	0.8514	0.8569	0.8363	0.8005	0.7204
Sensitivity	0.8698	0.8525	0.8773	0.9353	0.5534

Table 2. Average of scores across all three images

When smoothed with the same set of Gaussians, Laplacian Results in very few values past (0.1,0.1), this is because smoothing the image results in gradual changes of intensity. This is made more viable by applying much smaller Gaussians.

4 EVALUATION

As you can see Laplacian over Gaussian, there is a very small change in threshold required to cause for almost perfect TPR, however due to the high clustering of data near the origin, it is worse than the randomly guessing in the vast majority of cases. Due to these properties, this would make it a highly effective filter for an image that has minimal noise, but will struggle otherwise. The clustering of data can still be used if the point where it is no longer worse is calculated, the result can simply be inverted for these values.

The derivative filters on the other hand, while note having as high sensitivity, average a higher specificity, achieve a higher specificity; making them more effective in cases with more noise.

The Laplacian filter could be considered an adequate choice in both cases since it achieves mediocre scores in both respects, reducing the risk of false positives, but also that of true positives.

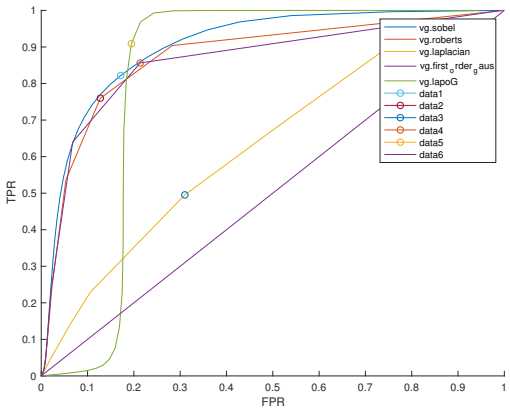


Figure 3. Optimal curves when tested with identical Gaussians of size 0-10 on '43590 AM.bmp'

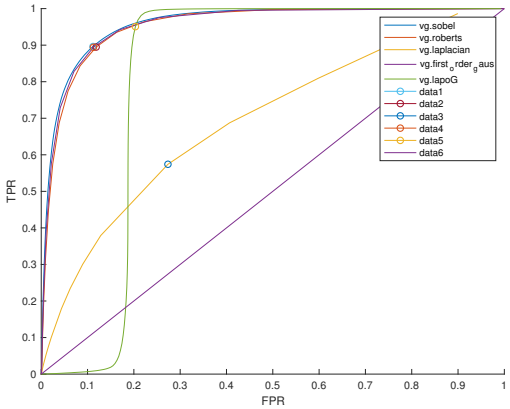


Figure 4. Optimal curves when tested with identical Gaussians of size 0-10 on '10905 JL.bmp'

REFERENCES

[1]https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm

[2]https://uk.mathworks.com/help/matlab/ref/del2.html