Edge Detection

Ziv Yaniv

School of Engineering and Computer Science
The Hebrew University, Jerusalem, Israel.

This lecture summary deals with the low level image processing task of edge detection. Edges are discontinuities, significant local changes, in image intensities which arise from three sources: (1)projection of 3D contours (2)texture present on the 3D surfaces (3)shadows cast by the imaged objects.

The summary includes the classical derivative based operators due to Canny [2], and Marr [7] and one non derivative based operator SUSAN [9].

We will not deal with the subject of algorithm evaluation. People which would like to read about this subject are referred to [1, 5, 8] evaluation studies of edge detection algorithms according to different criteria.

The summary is divided into three sections: (1) Derivative based operators. (2) The SUSAN edge detector. (3) Post processing.

1 Derivative Based Operators

At the basis of all derivative based operators is the following observation: edges appear as maxima of the first derivative and as zero crossings of the second derivative (Figure 1).

We now look at the approximation and application of derivatives to images.

1.1 Derivative Estimation

A derivative of a function $f(x)$ is given by:

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

An image is a bivariate function $I(x, y)$ and we want to estimate the partial derivative in the x and y directions. This is done using a discrete approximation of Equation 1:

$$\frac{\partial}{\partial x} = \begin{array}{|c|c|}\hline -1 & 1 \\ \hline\end{array} \qquad \frac{\partial}{\partial y} = \begin{array}{|c|}\hline -1 \\ \hline 1 \\ \hline\end{array}$$

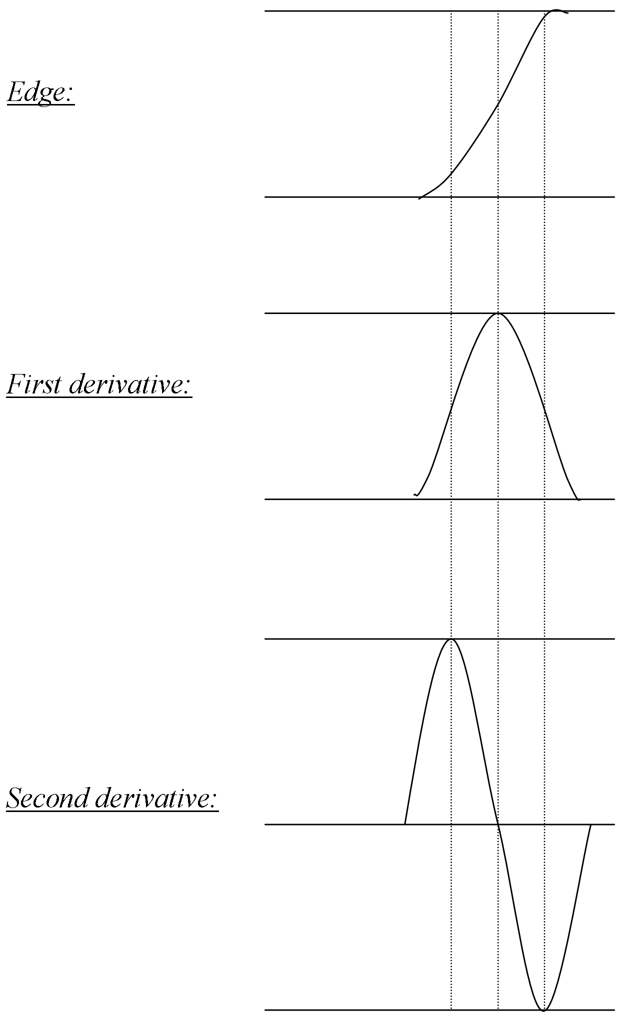


Figure 1: Original edge and its derivatives.

Unfortunately this operator does not give the approximation for the same point in space in the x and y directions. In the x direction the approximation is for the point $(x + 0.5, y)$ while for the y direction it is for the point $(x, y + 0.5)$

The solution to this problem is to use the following approximation:

$$\frac{\partial}{\partial x} = \begin{array}{|c|c|} \hline -1 & 1 \\ \hline -1 & 1 \\ \hline \end{array} \qquad \frac{\partial}{\partial y} = \begin{array}{|c|c|} \hline -1 & -1 \\ \hline 1 & 1 \\ \hline \end{array}$$

which approximates the derivative at the location $(x+0.5, y+0.5)$. A similar approximation is given by the Roberts differentiation kernel (Figure 2). The only difference is that the approximation is not along the x, y axis. Convolving the image with these partial derivative operators yields the image derivative. Unfortunately images contain noise and these operators are sensitive to noise, giving the same response to noise and signal. A standard solution to this problem is to smooth the image prior to differentiation yielding the following equations:

$$\frac{\partial I}{\partial x} = M_{\frac{\partial}{\partial x}} * S * I$$

$$\frac{\partial I}{\partial y} = M_{\frac{\partial}{\partial y}} * S * I$$

where S is a smoothing kernel and $M_{\frac{\partial}{\partial x}}, M_{\frac{\partial}{\partial y}}$ differentiation kernels in the x and y directions.

Two operators which apply this approach are the Prewitt and Sobel operators Figures 3 and 4 respectively. Where the Prewitt smoothing kernel is

1	1	1

and the Sobel smoothing kernel is

1	2	1

Both operators use the following difference operator for differentiation:

-1	0	1

Gradient direction is given by the partial derivatives $\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y}$:

$$\theta = arctan(\frac{\partial I(x,y)}{\partial y}, \frac{\partial I(x,y)}{\partial x})$$

Gradient magnitude is usually computed with one of the following formula:

$$\begin{aligned} \text{magnitude} &= \left|\frac{\partial I(x,y)}{\partial y}\right| + \left|\frac{\partial I(x,y)}{\partial x}\right| \\ \text{magnitude} &= \left(\frac{\partial I(x,y)}{\partial y}\right)^2 + \left(\frac{\partial I(x,y)}{\partial x}\right)^2 \\ \text{magnitude} &= \sqrt{\left(\frac{\partial I(x,y)}{\partial y}\right)^2 + \left(\frac{\partial I(x,y)}{\partial x}\right)^2} \end{aligned}$$

$\frac{\partial}{\partial x}$

1	0
0	-1

$\frac{\partial}{\partial y}$

0	-1
1	0

Figure 2: Roberts gradient estimation operator.

$\frac{\partial}{\partial x}$

-1	0	1
-1	0	1
-1	0	1

$\frac{\partial}{\partial y}$

-1	-1	-1
0	0	0
1	1	1

Figure 3: Prewitt gradient estimation operator.

$\frac{\partial}{\partial x}$

-1	0	1
-2	0	2
-1	0	1

$\frac{\partial}{\partial y}$

-1	-2	-1
0	0	0
1	2	1

Figure 4: Sobel gradient estimation operator.

1.2 Canny Edge Detector

This edge detector is due to J.F. Canny [2] (a recursive implementation of this algorithm was presented in [4]).

In his work Canny specified several criteria for the performance of edge detectors:

1. Minimum number of false negatives and false positives.
2. Good localization, report edge location at correct position.
3. Single response to single edge.

Solving an optimization problem using variational calculus and the criteria specified above he arrived at an optimal edge enhancing filter, the derivative of a Gaussian $G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$:

$$\frac{\partial G}{\partial x} = -\frac{x}{\sigma^2}G$$
$$\frac{\partial G}{\partial y} = -\frac{y}{\sigma^2}G$$

Further attributes of the Gaussian and its derivatives can be found in the appendix of this lecture.

Canny Edge Detection

1. Convolve the image with the derivative of a Gaussian.
2. Apply non maxima suppression to the gradient magnitude image.
3. Use two thresholds $\tau_1 > \tau_2$:
   (a) $\text{class} = \begin{cases} \text{edge} & \text{if magnitude} > \tau_1 \\ \text{candidate} & \text{if magnitude} > \tau_2 \end{cases}$
   (b) Hysteresis: Any candidate which is a neighbor, in the gradient direction, of an edge is reclassified as an edge.

Table 1: Canny Edge Detector

The steps of the edge detection process are as follows (summary in Table 1):

Start by convolving the image with the derivatives of a Gaussian mask. The result is a magnitude image with maxima at edge locations. Unfortunately this image may contain broad ridges at edge locations, which brings us to the next step.

Non maxima suppression is the process of thinning these ridges. At each pixel check if it is a local maxima in the gradient direction, if it is then retain it, otherwise change the magnitude to zero. The magnitude along the gradient direction is usually computed using linear interpolation as shown in Figure 5.

The third and final step of the algorithm uses a double thresholding scheme. A high threshold τ_1 is selected such that all pixels with gradient magnitude greater than τ_1 are classified as edge elements (edgels). A second threshold $\tau_2 < \tau_1$ is selected and all pixels with gradient magnitude greater than τ_2 are classified as candidate edgels. Now reclassify the candidates, if a candidates neighbor in the direction perpendicular to the gradient is an edgel then the candidate is reclassified as an edgel too. This reclassification is usually referred to as hysteresis [1]

1.2.1 Sub-Pixel Location Estimation

The approach described in this subsection is due to [3] (this is a standard approach for locating a local extremum).

Once a pixel is determined as an edge we look at the original gradient magnitude data. Again we have three magnitude values as shown in Figure 5. We will fit a parabola to these values and find its maxima which is our sub pixel estimate for the edge location. Given three points $(-1, v_1)$, $(0, v_2)$, $(1, v_3)$ we want to fit a parabola, $f(x) = a_1x^2 + a_2x + a_3$,

[1]Hysteresis: The phenomenon exhibited by a system, often a ferromagnetic or imperfectly elastic material, in which the reaction of the system to changes is dependent upon its past reactions to change (Websters Encyclopedic Unabridged Dictionary).

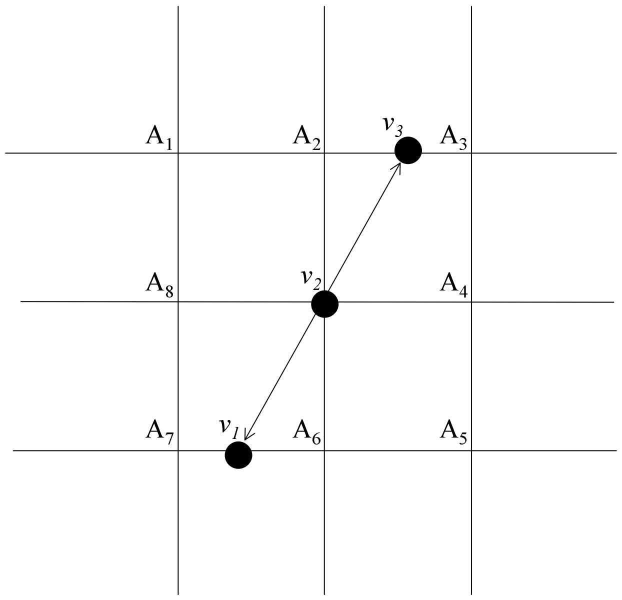


Figure 5: Non Maximal Suppression. The pixel with gradient magnitude v_2 is retained only if $v_2 > v_1$ and $v_2 > v_3$. The values at v_1 and v_3 are computed using linear interpolation between neighboring pixels.

through these points. Solving the following three equations yields the unknown coefficients:

$$\begin{aligned}
&a_1 - a_2 + a_3 = v_1 \\
&a_3 = v_2 \\
&a_1 + a_2 + a_3 = v_3
\end{aligned}$$

The maximum of the parabola is given for:

$$f'(x) = 0, \; x_{max} = -\frac{a_2}{2a_1}$$

Substituting the computed coefficients into the equation above yields:

$$x_{max} = \frac{v_1 - v_3}{2(v_1 + v_3 - 2v_2)}$$

which is the distance along the gradient direction in which the edge is located.

1.3 Caveats

Algorithms which smooth the image prior to differentiation will cause errors in edge location. More smoothing means that detected edges will be farther away from their actual location.

Algorithms which apply non maxima suppression rely on the gradient direction for this operation. When smoothing is applied to contours with high curvature the gradient

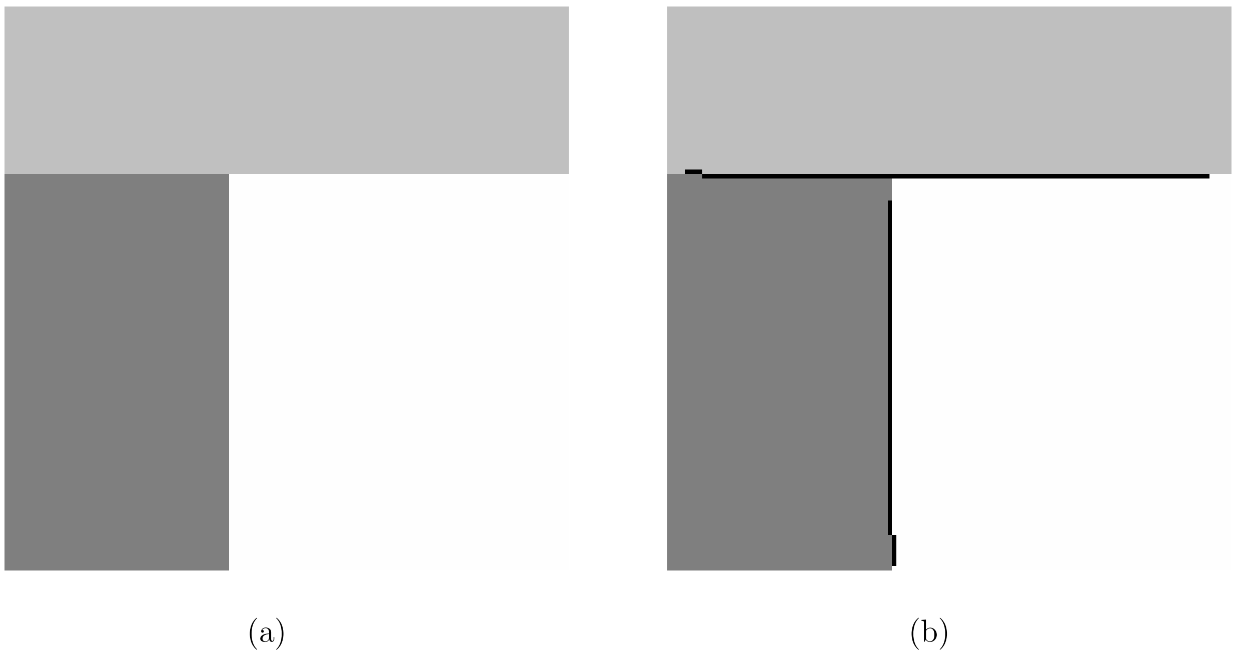


Figure 6: (a) Original and (b) result of Canny edge detector ('edge' function in matlab) marked in black.

direction is usually not correct. At corners this will usually yield wrong directions. This is the reason why the Canny edge detector gives broken edges at T-junctions as shown in Figure 6.

1.4 Marr-Hildreth (LoG) Edge Detector

LoG Edge Detection

1. Convolve the image with the Laplacian of a Gaussian.
2. Find zero crossings. Usually there will not be pixels whose value is zero. Given a pixel with value v_1 look at its neighbors. If there is a neighbor with value v_2 such that $v_1 v_2 < 0$ then if $|v_1| < |v_2|$ mark the pixel as an edge.

Table 2: LoG Edge Detector

At the beginning of this section we saw that edges correspond to maxima of the first derivative and zero crossings of the second derivative. The edge detector described in this subsection is based on the later observation and is due to [7].

The algorithm applies a smoothing filter, a Gaussian, to the original image, computes the laplacian of the smoothed image and searches for zero crossings in it. The laplacian of a bivariate function $I(x, y)$ is given by:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

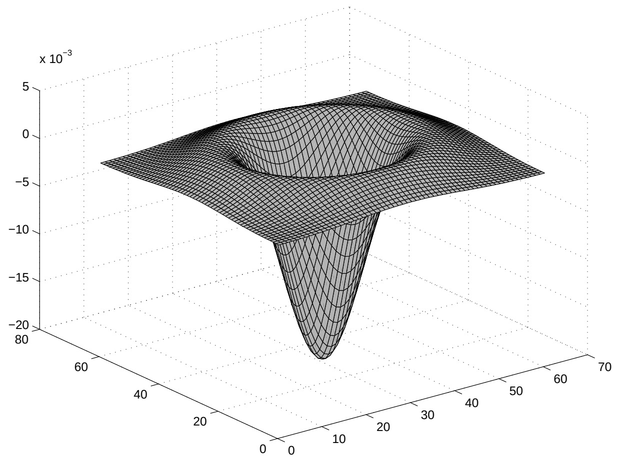


Figure 7: Two dimensional laplacian of Gaussian.

The output of the Laplacian of Gaussian (LoG), operator on the image $I(x, y)$ using Gaussian $G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$ is given by:

$$\begin{gathered} LoG(I(x, y)) = \nabla^2[G(x, y) * I(x, y)] \\ \Downarrow \\ LoG(I(x, y)) = [\nabla^2 G(x, y)] * I(x, y) \\ \Downarrow \\ LoG(I(x, y)) = [\left(\tfrac{x^2+y^2-2\sigma^2}{\sigma^4}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}] * I(x, y) \end{gathered}$$

The image $LoG(I(x, y))$ is searched for zero crossings which are rarely located at pixel locations. What we search for are pixels with different signs and the edge is located between them. Using linear interpolation between two such pixels yields the edge location with sub pixel accuracy.

The LoG operator can be approximated using the following mask:

0	1	0
1	-4	1
0	1	0

An important thing to note is that with the LoG operator no directional data is available. This should be kept in mind if the higher level algorithms which receive the output of the edge detection require edge direction.

SUSAN Edge Detection

```
1. For each pixel do:
    (a) Calculate the USAN (Equation 2).
    (b) Calculate edge response according to Equation 3.
2. Apply non maxima suppression to the edge response image (using moment
   calculations on the USAN).
3. Apply thinning (post processing).
```

Table 3: SUSAN Edge Detector

2 The SUSAN edge detector

This edge detector is a non derivative based operator due to S.M. Smith [9] [2].

SUSAN stands for "Smallest Univalue Segment Assimilating Nucleus". The idea behind this detector is to use a pixel's similarity to its neighbors gray values as the classification criteria (a non linear filter). In Figure 8 we see that the area of the USAN contains the information about the image structure around a given point which allows classification. The area of the USAN is at a maximum in a flat region, falls to half near a straight edge and falls further when the mask is placed near a corner.

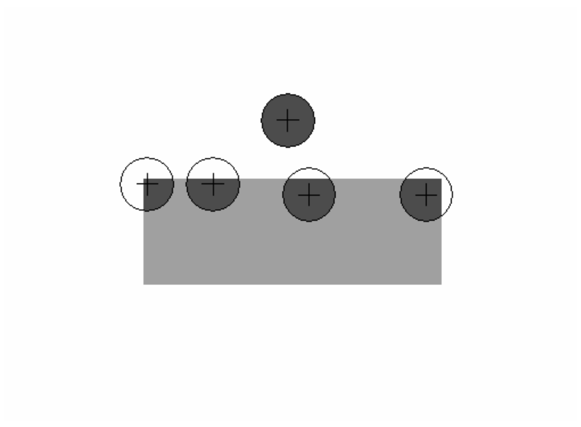


Figure 8: Univalue Segment Assimilating Nucleus (USAN). Circular masks placed at different locations of an image containing a single rectangle. The USAN of each mask is marked in dark color. The nucleus of the masks are marked with a +.

[2] This detector is patented (UK Patent 2272285)

The steps of the edge detection are as follows (summary in Table 3):

At each pixel place a circular mask and compute the weight of the USAN. The weight of the USAN is:

$$n(r_0) = \sum_r \text{compare}(r, r_0) \tag{2}$$

where compare(r, r_0) was originally defined as:

$$\text{compare}(r, r_o) = \begin{cases} 1 & if|I(r) - I(r_0)| \leq t \\ 0 & if|I(r) - I(r_0)| > t \end{cases}$$

with t a threshold defining pixel gray level similarity. The comparison function actually used is a "smoother" version of the original:

$$\text{compare}(r, r_0) = e^{-(\frac{I(r)-I(r_0)}{t})^6}$$

Compute the edge strength at the pixel using the following formula:

$$\text{response}(r_o) = \begin{cases} g - n(r_0) & \text{if } n(r_0) < g \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

with g a geometric threshold which is set to $\frac{3}{4}n_{max}$ (proof of optimality of this threshold can be found in [9]).

Having computed the edge response image we now perform non maxima suppression (NMS). NMS requires the direction perpendicular to the edge. The direction depends on the edge type which is being examined either inter-pixel (edge is between pixels) or intra-pixel (pixel itself is part of the edge).

Inter pixel case, if the USAN area is greater than the mask diameter and the center of gravity of the USAN lies more than one pixel from the nucleus. The center of gravity of the USAN is defined as:

$$CG(r_0) = \frac{\sum_r \text{r compare}(r, r_0)}{\sum_r \text{compare}(r, r_0)}$$

The direction we want is given by $(r_0 - CG(r_0)$.

Intra pixel case, if the USAN area is smaller than the mask diameter or the USAN center of gravity lies less than one pixel from the nucleus. Compute the second order moments of the USAN about the nucleus ($r_0 = (x_0, y_0)$):

$$\overline{(x - x_0)^2)} = \sum_r (x - x_0)^2 \text{compare}(r, r_0)$$
$$\overline{(y - y_0)^2)} = \sum_r (y - y_0)^2 \text{compare}(r, r_0)$$

Edge orientation is given by $\frac{\overline{(y-y_0)^2)}}{\overline{(x-x_0)^2)}}$.

A nice attribute of the SUSAN edge detector is that it can handle T-junctions well (Figure 9) as it does not rely on gradient direction for NMS which was a problem with the Canny detector (Figure 6).

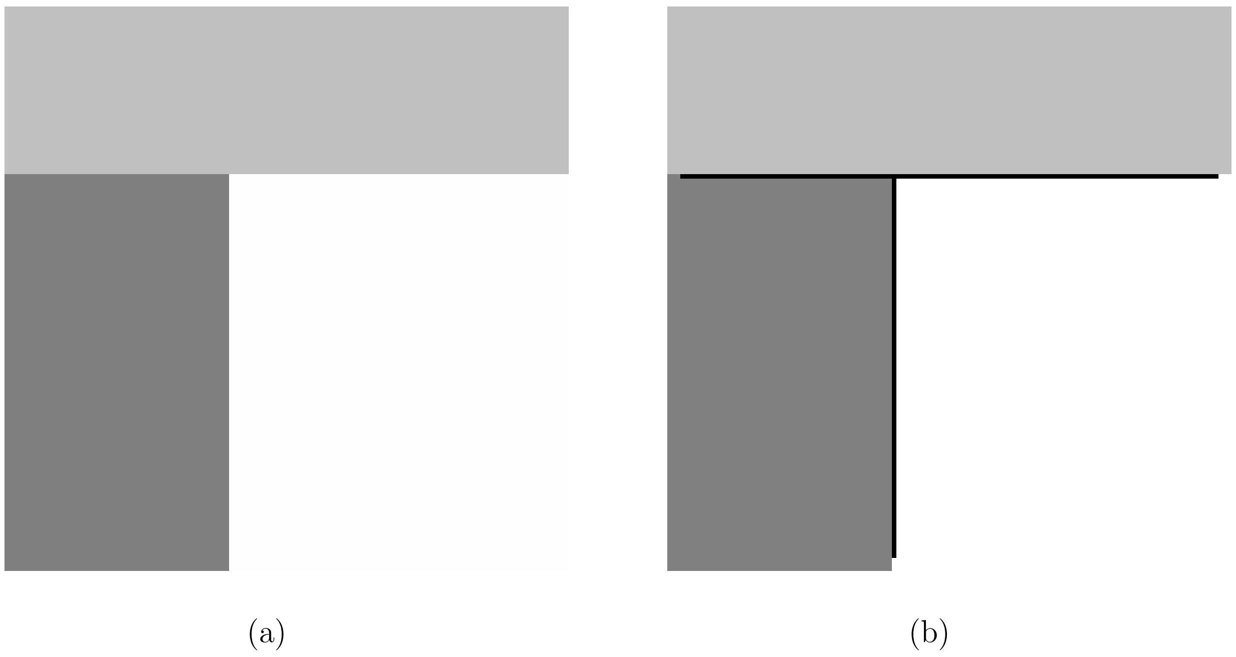


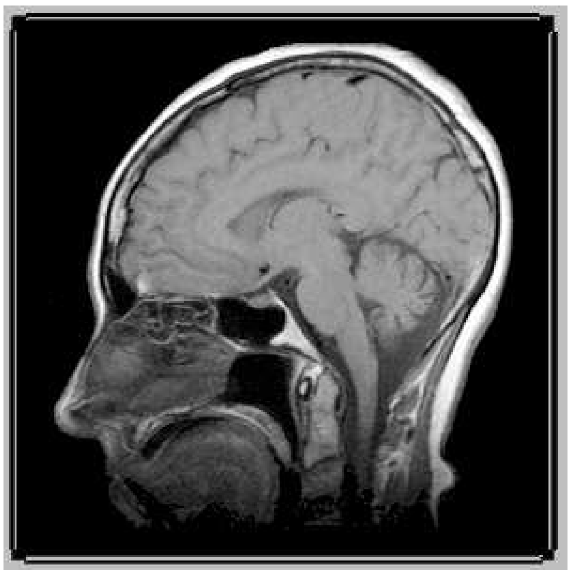
Figure 9: (a) Original and (b) result of SUSAN edge detector marked in black.

3 Post Processing

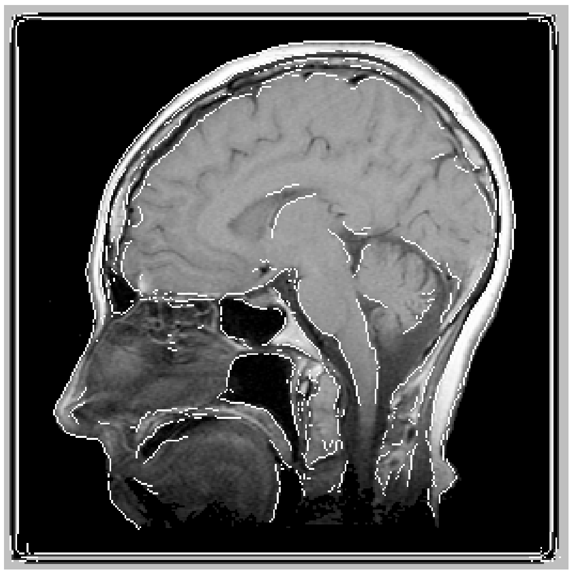
Many implementations of edge detectors apply post processing steps to remove noise and thin edges.

Thining is usually done with morphological operators which are not discussed here. This post processing seems unnecasery if the operator includes a non maxima suppresion (NMS) step, but in practice NMS does not always remove multiple responses to the same edge, due to inaccuracies in gradient direction.

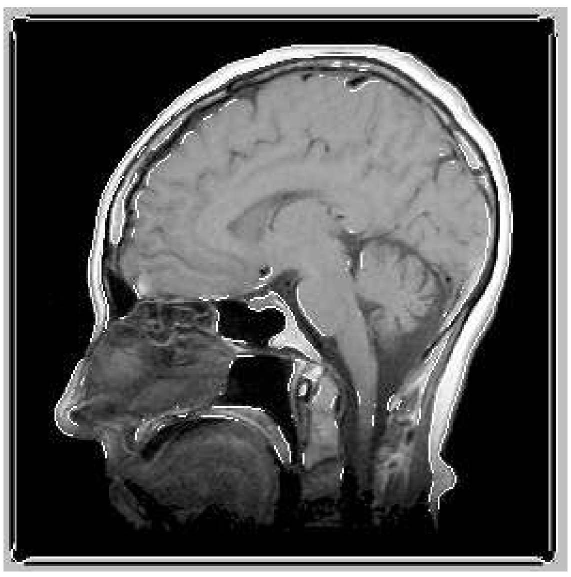
Another heuristic is to remove small connected components which are assumed to be the result of noise in the image. The definition of small is vague, remaining an implementation issue.



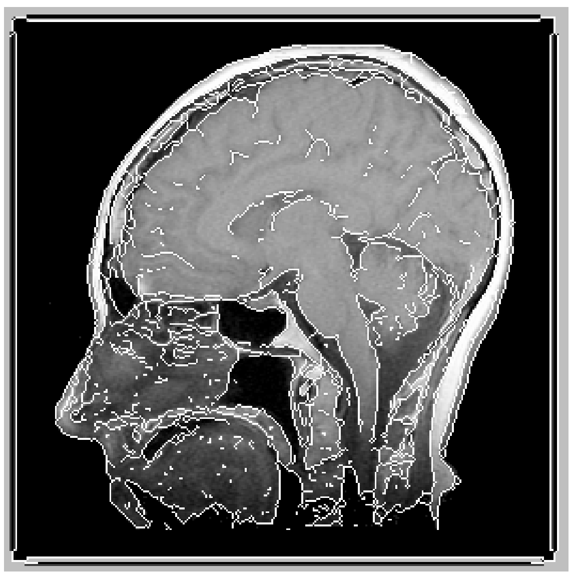
(a)



(b)



(c)



(d)

Figure 10: (a) Original (b) result of Canny edge detector with $\sigma = 3$ (c) result of LoG edge detector with $\sigma = 3$ (d) result of SUSAN edge detector

A The Gaussain Kernel

A common smoothing kernel is the zero mean Gaussian which is given by the following equation:

$$G(x,y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

This kernel has several nice attributes:

- Rotational symmetry. For unisotropic smoothing use a Gaussian kernel with different standard deviations in the x and y directions.
- Its Fourier transform is also a Gaussian.
- Seperability.

The seperability leads to faster convolution using two 1D kernels:

$$\begin{aligned}
G(x,y) * I(x,y) &= \textstyle\sum_{i=1}^{m}\sum_{j=1}^{n} G(i,j)I(x-i,y-j) \\
&= \textstyle\sum_{i=1}^{m}\sum_{j=1}^{n} e^{\frac{-(i^2+j^2)}{2\sigma^2}} I(x-i,y-j) \\
&= \textstyle\sum_{i=1}^{m} e^{\frac{-i^2}{2\sigma^2}} [\sum_{j=1}^{n} e^{\frac{-j^2}{2\sigma^2}} I(x-i,y-j)] \\
&= G(x) * G(y) * I(x,y)
\end{aligned}$$

Approximating the Gaussian:

A good approximation to a Gaussian is given by the coefficients of the binomial expansion:

$$(1+x)^n = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n$$

This gives Pascal's Triangle:

```
         1
      1     1
   1     2     1
1     3     3     1
```

Convolution with a Gaussian is separable so we can use rows from Pascal's triangle as smoothing kernels, for example:

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * I(x,y)$$

This approach is not always directly applicable as the binomial coefficients may be too large for representation on the computer. The same effect is achieved by repeatedly convolving with smaller filters.

Gaussian smoothing and image differentiation:
The following formula describes the use of Guassian smoothing prior to differentiation in the x direction (symmetric for the y direction):

$$\begin{aligned}\frac{\partial}{\partial x}(G(x,y) * I(x,y)) &= (\frac{\partial G(x,y)}{\partial x}) * I(x,y) \\ &= (-\frac{x}{\sigma^2}G(x,y)) * I(x,y) \\ &= -\frac{x}{\sigma^2}G(x) * G(y) * I(x,y) \\ &= \frac{dG(x)}{dx} * G(y) * I(x,y)\end{aligned}$$

References

[1] Bowyer K.W., Kranenburg C., Dougherty S. "Edge Detector Evaluation Using Empirical ROC Curves" IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 354-359, 1999.

[2] Canny J.F., "A computational approach to edge detection", IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE TPAMI), Vol. 8(6), pp. 769–798, 1986.

[3] Devernay F., "A Non-Maxima Suppression Method for Edge Detection with Sub-Pixel Accuracy", Research report 2724, INRIA Sophia-Antipolis, 1995.

[4] Deriche R., "Using canny's criteria to derive a recursively implemented optimal edge detector", International Journal of Computer Vision (IJCV), Vol. 1(2), pp. 167–187, 1987.

[5] Heath M., Sarkar S., Sanocki T., Bowyer K.W. "A Robust Visual Method for Assessing the Relative Performance of Edge Detection Algorithms", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol.19(12), pp. 1338-1359, 1997.

[6] Jain R., Kasturi R., and Schunk B.G., *Machine Vision*, McGraw-Hill, 1995.

[7] Marr D., Hildreth E., "Theory of Edge Detection", Proceedings of Royal Society of London, Vol. 207, pp. 187–217, 1980.

[8] Shin M., Goldgof D., Bowyer K.W., "An Objective Comparison Methodology of Edge Detection Algorithms for Structure from Motion Task", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 190-195, 1998.

[9] Smith S.M., Brady J.M., "SUSAN - a new approach to low level image processing", International Journal of Computer Vision (IJCV), Vol. 23(1), pp. 45–78, 1997.