Kyungpook National University

Artificial Brain Research - ABR

BEP

Lesson and Assignment 6

Edge

Gwenaelle Cunha Sergio

Winter 2014

1 Summary of Lecture Material

Filtering, edge detection and morphology are all image pre-processing methods. The first one removes image noises, the second one is used to find boundaries or contours in an object and the third one is used to increase or decrease the relation between two parts of an object. They all use similar methods, or mask, and ways (convolution).

An image convolution is represented by the following formula, where h is the filter, $*$ denotes the convolution and m is the filter size:

$$p(x,y) = q * h = \sum_{i=-m/2}^{m/2} \sum_{j=-m/2}^{m/2} h(j,i)q(x-j,y-i)$$

1.1 Edge Detection

An edge is detected when there's a change in intensity, or sharp gradient, in the image, as seen in **Fig. 1**.

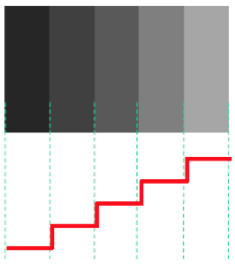


Figure 1: Edge and gradient

Normalization is used to make sure that the values stay between 0 and 255, the color range in the RGB color space:

$$\frac{currentValue - min}{max - min} * value$$

1.1.1 Methods

Sobel: image can be analyzes on the x and/or y axis, see **Fig. 2**.

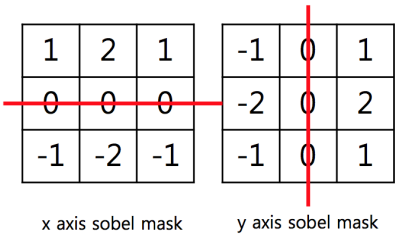
Prewitt: very similar to Sobel, only a few values change in the mask, see **Fig. 3**.

Roberts: it's faster than Sobel and Prewitt and smaller than other masks, so emphasized pixels can't be averaged, that is, it's sensitive to noises. See **Fig. 4**.

Gaussian: it's a 2 order differential edge detection that reflects the distribution of a Gaussian Curve, **Fig. 5(a)**.

Laplacian: like the Gaussian method, the Laplacian one is also a 2 order differential edge detection method. Edges can be detected in any direction and noise is small. See **Fig. 6**.

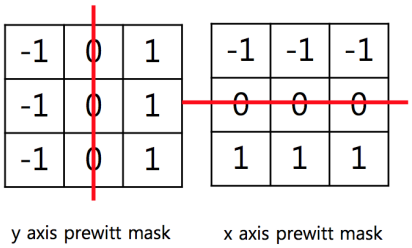
Laplacian of Gaussian (LoG): it's also a 2 order differential edge detection method. The Gaussian Operator is obtained by differentiating Laplacian Operator.



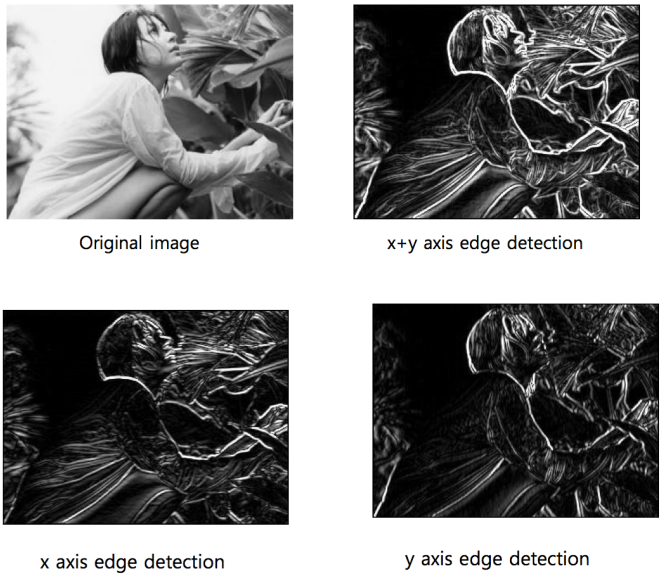
(a) Sobel Mask

(b) Result image for Sobel Mask

Figure 2: Sobel

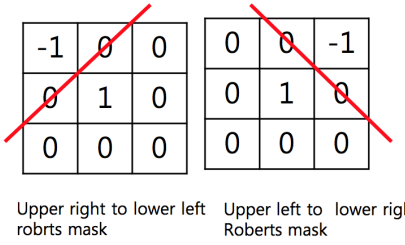


(a) Prewitt Mask

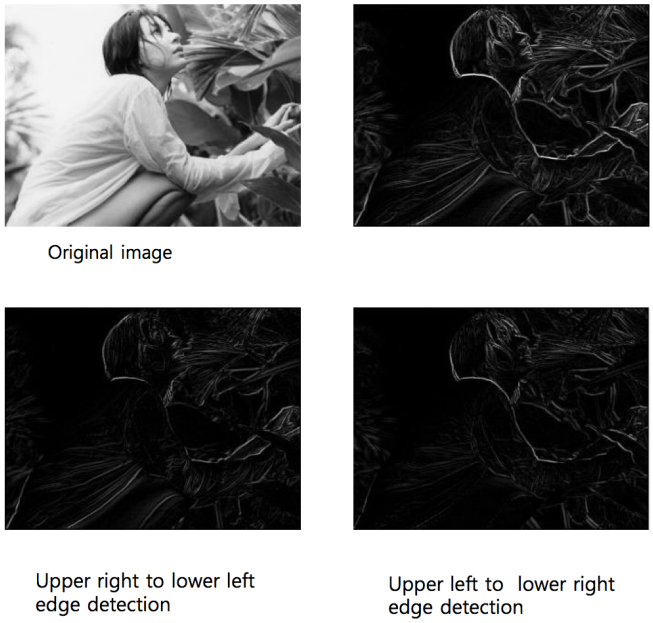


(b) Result image for Prewitt Mask

Figure 3: Prewitt

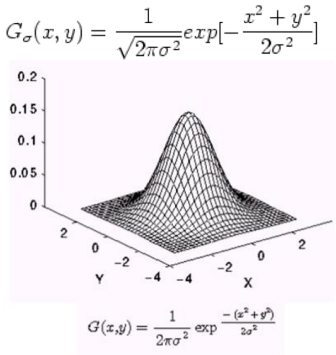


(a) Roberts Mask



(b) Result image for Roberts Mask

Figure 4: Roberts



$$\frac{1}{159} \times \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

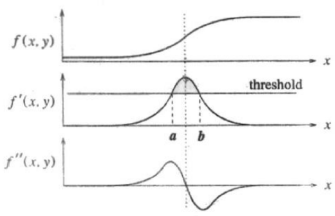
(a) Gaussian Mask

(b) Result image for Gaussian Mask

Figure 5: Gaussian

1	1	1
1	-8	1
1	1	1

0	1	0
1	-4	1
0	1	0



(a) Laplacian Mask

(b) Result image for Laplacian Mask

Figure 6: Laplacian

Difference of Gaussian (DoG): it's also a 2 order differential edge detection method. Uses 2 different σ as Gaussian Operator to detect edge, **Fig. 7**. It's less time consuming.

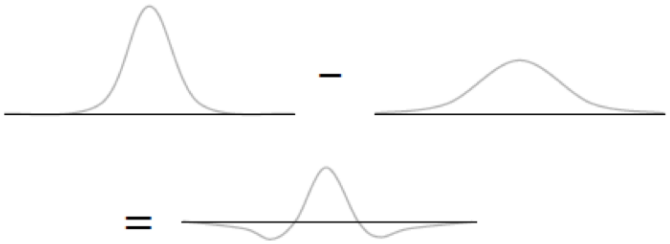
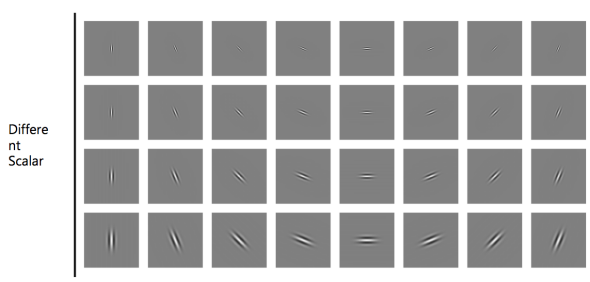


Figure 7: DoG

Gabor: detects edge orientation. See **Fig. 8**.



Gabor filter select necessary orientation edge

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

90^0 orientation

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

180^0 orientation

Figure 8: Gabor Filter

Highboost-spatial: **Fig. 9** makes it much more intuitive to understand this method. It's basically a high-pass filter with a boost, like the low-pass one. See **Fig. 10**.

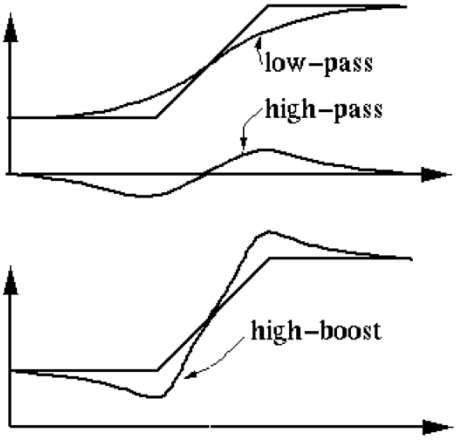
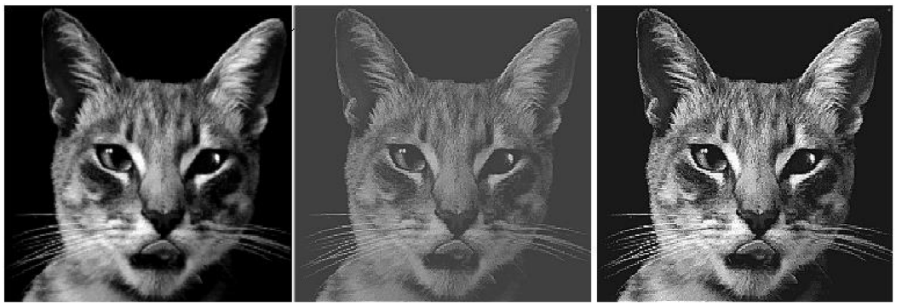


Figure 9: Highboost Graph

$$W_{highboost} = cW_{allpass} + W_{highpass} = c\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4+c & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$W_{highboost} = cW_{allpass} + W_{highpass} = c\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8+c & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

(a) Highboost Mask



(b) Result image for Laplacian Mask

Figure 10: Highboost

2 Assignment

This assignment consists of answering the following questions.

2.1 Edge? How to use?

An edge is a part of the image that has a sharp gradient (change in intensity/contrast). Its detection can be used to extract information about the image, like location of objects present in the image, their size, shape, image sharpening and enhancement [1].

Applications of edge detection are: enhancement of noisy images such as satellite images, x-rays and medical images like cat scans; text detection; mapping of roads; video surveillance; and remote sensing images.

2.2 Implement Edge

Like in the average filter, this one can be implemented using the *filter2D* function, but with a different mask. For 3x3:

```
kernel.at<float>(0,0) = 1; kernel.at<float>(0,1) = 2; kernel.at<float>(0,2) = 1;
kernel.at<float>(1,0) = 0; kernel.at<float>(1,1) = 0; kernel.at<float>(1,2) = 0;
kernel.at<float>(2,0) = -1; kernel.at<float>(2,1) = -2; kernel.at<float>(2,2) = -1;
filter2D(mat, filter3x3, ddepth, kernel, anchor, delta, BORDER_DEFAULT);
```

2.3 Change Mask size and implement, analysis the results

Figure 11: Edge Detection with Sobel: original, 3x3 and 5x5

2.4 Implement Sharpening(Using Highboost-spatial)

Two kernels are going to be used to analyze this filter. Consider $c = 1$.

[1] $http://www.slideshare.net/contactsarbjeet/image-processing-edge-detection$

```
kernel = Mat::zeros(kernel_size, kernel_size, CV_32F);
kernel.at<float>(0,1) = -1; kernel.at<float>(1,0) = -1;
kernel.at<float>(1,2) = -1; kernel.at<float>(1,1) = 4+c;
kernel.at<float>(2,1) = -1;
filter2D(mat, edge, ddepth, kernel, anchor, delta, BORDER_DEFAULT);

kernel = -1*Mat::ones(kernel_size, kernel_size, CV_32F);
kernel.at<float>(1,1) = 8+c;
filter2D(mat, edge2, ddepth, kernel, anchor, delta, BORDER_DEFAULT);
```

Figure 12: Highboost-spatial: original, kernel1 and kernel2

2.5 4 direction, implement 0, 45, 90, 135 Edge Orientation

First, we open the image and set the variables:

```
cv::Mat mat = imread("TestEdge.png", CV_LOAD_IMAGE_GRAYSCALE);
Mat kernel, edgeX, edgeY, edge0, edge45, edge90, edge135;
mat.copyTo(edgeX); mat.copyTo(edgeY);
edge0 = Mat::zeros(mat.rows, mat.cols, mat.type());
edge45 = Mat::zeros(mat.rows, mat.cols, mat.type());
edge90 = Mat::zeros(mat.rows, mat.cols, mat.type());
edge135 = Mat::zeros(mat.rows, mat.cols, mat.type());

cv::Point anchor = cv::Point(-1,-1);
double delta = 0;
int ddepth = -1;
int kernel_size = 3;
```

Then, we calculate Sobel on the x and y axis:

```
kernel = Mat::ones(kernel_size, kernel_size, CV_32F);
kernel.at<float>(0,0) = 1; kernel.at<float>(0,1) = 2; kernel.at<float>(0,2) = 1;
kernel.at<float>(1,0) = 0; kernel.at<float>(1,1) = 0; kernel.at<float>(1,2) = 0;
```

Figure 13: Edge orientation original

```
kernel.at<float>(2,0) = -1; kernel.at<float>(2,1) = -2; kernel.at<float>(2,2) = -1;
filter2D(mat, edgeX, ddepth, kernel, anchor, delta, BORDER_DEFAULT);

kernel = Mat::ones(kernel_size, kernel_size, CV_32F);
kernel.at<float>(0,0) = -1; kernel.at<float>(0,1) = 0; kernel.at<float>(0,2) = 1;
kernel.at<float>(1,0) = -2; kernel.at<float>(1,1) = 0; kernel.at<float>(1,2) = 2;
kernel.at<float>(2,0) = -1; kernel.at<float>(2,1) = 0; kernel.at<float>(2,2) = 1;
filter2D(mat, edgeY, ddepth, kernel, anchor, delta, BORDER_DEFAULT);
```

Lastly, we calculate the arctangent, *atan2*, using the data just obtained, resulting in **Fig. 14**:

```
for (int i = 0; i < mat.rows; i++){
    for (int j = 0; j < mat.cols; j++){
        uchar val = atan2((double) edgeY.at<uchar>(i,j),
            (double) edgeX.at<uchar>(i,j))*180 / CV_PI;
        if (val < 45) edge0.at<uchar>(i,j) = val;
        else if (val < 90) edge45.at<uchar>(i,j) = val;
        else if (val < 135) edge90.at<uchar>(i,j) = val;
        else edge135.at<uchar>(i,j) = val;
    }
}
```

2.6 Laplacian, LoG, DoG formula

It's important to know that I(x,y) is the image and that the formula of a Gaussian is:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = Ce^{-\frac{x^2+y^2}{2\sigma^2}}$$

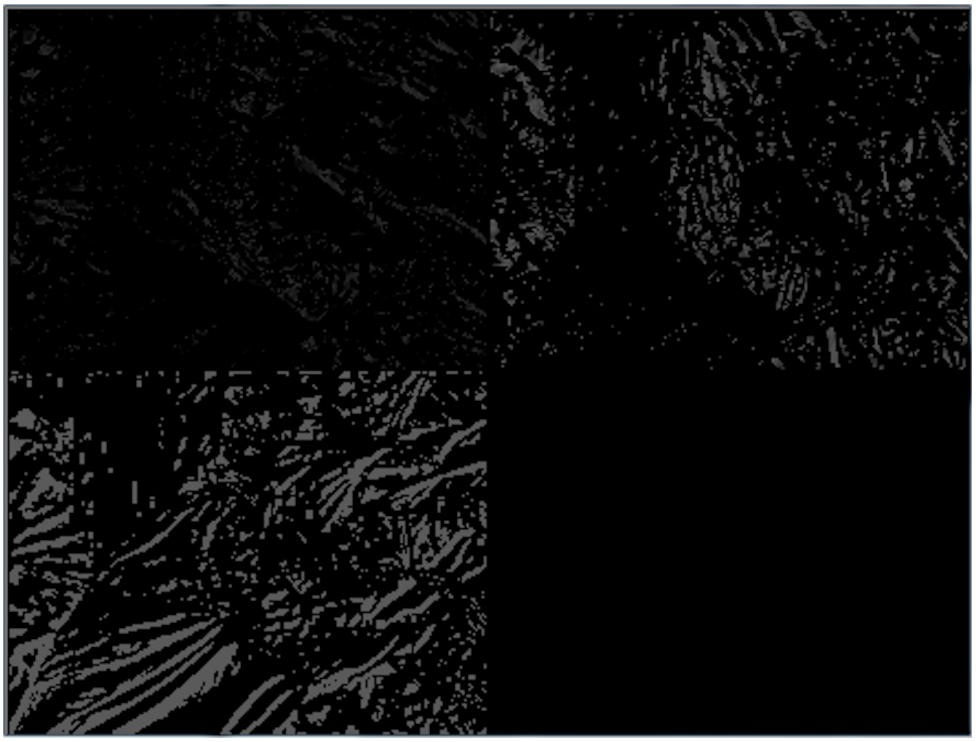


Figure 14: Edge orientation: 0, 45, 90 and 135 degrees

2.6.1 Laplacian

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

2.6.2 LoG

The Gaussian's first derivative for x is:

$$\begin{aligned} G'_x(x,y) &= \frac{\partial}{\partial x} G(x,y) \\ &= G(x,y) \frac{\partial}{\partial x} \left(-\frac{x^2+y^2}{2\sigma^2} \right) \\ &= -G(x,y) \frac{x}{\sigma^2} \end{aligned}$$

The Gaussian's first derivative for y is:

$$\begin{aligned} G'_y(x,y) &= \frac{\partial}{\partial y} G(x,y) \\ &= G(x,y) \frac{\partial}{\partial y} \left(-\frac{x^2+y^2}{2\sigma^2} \right) \\ &= -G(x,y) \frac{y}{\sigma^2} \end{aligned}$$

LoG, or Marr-Hildreth Edge Detector:

$$
\begin{aligned}
LoG(x,y) &= \Delta G(x,y) = \nabla^2 G(x,y) \\
&= \frac{\partial^2}{\partial x^2} G(x,y) + \frac{\partial^2}{\partial y^2} G(x,y) \\
&= \frac{\partial}{\partial x} G'_x(x,y) + \frac{\partial}{\partial y} G'_y(x,y) \\
&= \frac{\partial}{\partial x}\left[-G(x,y)\frac{x}{\sigma^2}\right] + \frac{\partial}{\partial y}\left[-G(x,y)\frac{y}{\sigma^2}\right] \\
&= -\frac{1}{\sigma^2}\left[\frac{\partial}{\partial x} xG(x,y) + \frac{\partial}{\partial y} yG(x,y)\right] \\
&= -\frac{1}{\sigma^2}\left[\frac{\partial}{\partial x}\left(xCe^{-\frac{x^2+y^2}{2\sigma^2}}\right) + \frac{\partial}{\partial y}\left(yCe^{-\frac{x^2+y^2}{2\sigma^2}}\right)\right] \\
&= -\frac{C}{\sigma^2}\left[\frac{\partial}{\partial x}\left(xe^{-\frac{x^2+y^2}{2\sigma^2}}\right) + \frac{\partial}{\partial y}\left(ye^{-\frac{x^2+y^2}{2\sigma^2}}\right)\right] \\
&= -\frac{C}{\sigma^2}\left[\left(e^{-\frac{x^2+y^2}{2\sigma^2}} + x\left(-\frac{x}{\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}\right) + \left(e^{-\frac{x^2+y^2}{2\sigma^2}} + y\left(-\frac{y}{\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}\right)\right] \\
&= -\frac{Ce^{-\frac{x^2+y^2}{2\sigma^2}}}{\sigma^2}\left[1 - \frac{x^2}{\sigma^2} + 1 - \frac{y^2}{\sigma^2}\right] \\
&= -\frac{1}{2\pi\sigma^4}\left[2 - \frac{x^2+y^2}{\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}} \\
LoG(x,y) &= \frac{1}{\pi\sigma^4}\left(\frac{x^2+y^2}{2\sigma^2} - 1\right)e^{-\frac{x^2+y^2}{2\sigma^2}}
\end{aligned}
$$

2.6.3 DoG

$$
\begin{aligned}
DoG(x,y) &= G_{\sigma_1} - G_{\sigma_2} \\
&= \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}} \\
&= \frac{1}{2\pi}\left[\frac{1}{\sigma_1} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{\sigma_2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}\right]
\end{aligned}
$$

2.7 Investigate Canny Edge

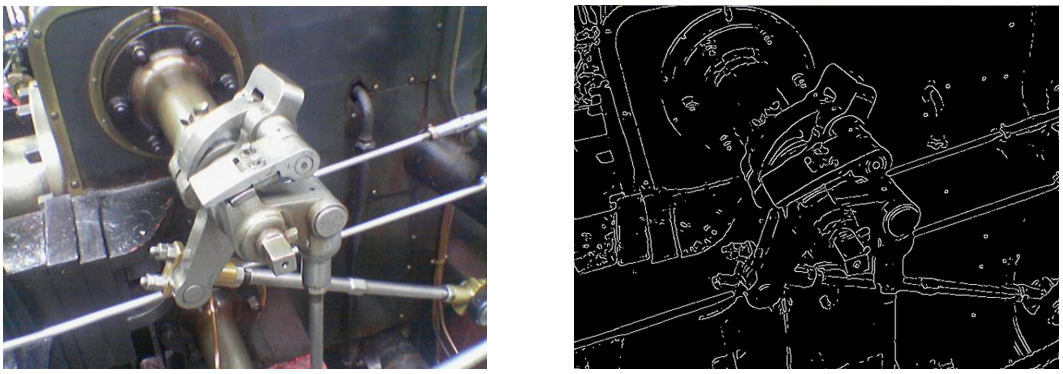
This method was developed by John F. Canny in 1986 and it's used to detect edges in an image. Canny's intention "was to discover the optimal edge detection algorithm" [2], meaning:

- Good detection: detect as many real edges in the image as possible;
- Good localization: edges marked should be as close as possible to the real edge;

[2] *http://en.wikipedia.org/wiki/Canny_edge_detector*

- Minimal response: an edge in the image should only be marked once, and image noise should not create false edges.

To satisfy these requirements Canny used a technique which finds the function which optimizes a given function (calculus of variations). The optimal function he found is described by the sum of four exponential terms that can be approximated by the first derivative of a Gaussian. The result of applying the Canny Filter in an image is shown in **Fig. 15**.



(a) Original Image (b) Result image using a Canny Mask

Figure 15: Canny

Feature detections methods (very interesting extra information):

- **Edge detection**: Canny, Canny Deriche, Differential, Sobel, Prewitt, Roberts cross;
- **Corner detection**: Harris operator, Shi and Tomasi, Level curve curvature, SUSAN, FAST;
- **Blob detection**: LoG, DoG, Determinant of Hessian (DoH), Maximally stable extremal regions, PCBR;
- **Affine invariant feature detection**: Affine shape adaptation, Harris affine, Hessian affine;
- **Feature description**: SIFT, SURF, GLOH, HOG;
- **Scale space**: Scale-space axioms, Implementation details, Pyramids.