Comparative Analysis of Sobel and Canny Techniques used for Detecting Edges that is a Part of Image Intensification

Kirti

M. Tech Student in C.E., IIET (Jind) under KUK, India
kirtipruthijind@gmail.com

Abhishek Bhatnagar

Asstt. Prof. in C.S.E., IIET (Jind) under KUK, India
ap.abhi.iiet@gmail.com

Abstract

Edges are defined as boundaries between different textures in an image. Or it may be defined as discontinuities in the intensity of the image from one pixel to another pixel. Edge detection is a part of image segmentation (i.e., partitioning the image into meaningful regions). Edge detection is used mainly for the purpose of storing large amount of information/data in a very small amount of memory area so that large amount of memory can be saved for other important tasks. As we know, the complete image uses a very large amount of memory and comparatively the edges use only a small part of that memory with complete information about that image. In this paper, the main focus is on comparative study of two different edge detection techniques used for image segmentation. These two edge detection techniques involved in this paper are namely sobel edge detection technique and canny edge detection technique.

Keywords: Edge Detection, Canny Edge Detection, Sobel Edge Detection, PSNR.

Introduction to Edge Detection

The points at which the brightness of an image changes suddenly are typically organized into a set of bent line segments termed as edges. Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction. The purpose of detecting sharp discontinuities or changes in the brightness of image is to capture important events and changes in the properties of the world. The discontinuities in an image may be of any kind. Some examples are:

- Discontinuities in depth.
- Discontinuities in surface orientation.
- Changes in material properties.

- Variations in scene illumination.

When we apply an image detection algorithm to an image, it may significantly reduce the amount of data to be processed and may therefore strain out information that may be regarded as less appropriate, while maintaining the important structural properties of an image. If the edge detection step is successful, the subsequent task of interpreting the information contents in the original image may therefore be substantially simplified. Edge detection is one of the most basic steps in image processing, image scanning, image pattern inspection, and computer vision techniques. Some of the commonly used edge detection algorithms are:

- Roberts Edge Detection Algorithm.
- Prewitt Edge Detection Algorithm.
- Fuzzy Logic Edge Detection Algorithm.
- Sobel Edge Detection Algorithm.
- Canny Edge Detection Algorithm.

The main focus of our research is on the comparative study and analysis of sobel edge detection algorithm and canny edge detection algorithm.

Sobel Edge Detection Technique

Generally edges are of three types:

- Horizontal edges.
- Vertical edges.
- Diagonal edges.

Sobel operator is a derivative operator that is used for detecting edges in horizontal and vertical directions. In sobel operator the coefficient of masks are not fixed and they can be adjusted according to our requirement unless they do not violate any property of derivative masks.

Vertical Mask of Sobel Operator

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

How it Works

When we apply this mask on the image, it prominent vertical edges. It simply works like as first order derivate and calculates the difference of pixel intensities in an edge region

As the center column is of zero, so it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge. Also the middle row values of both the first and third column is 2 and -2 respectively.

This give more weight age to the pixel values around the edge region. This raise the intensity of edge and it increase the strength of the image as compared to the original image.

Horizontal Mask of Sobel Operator

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

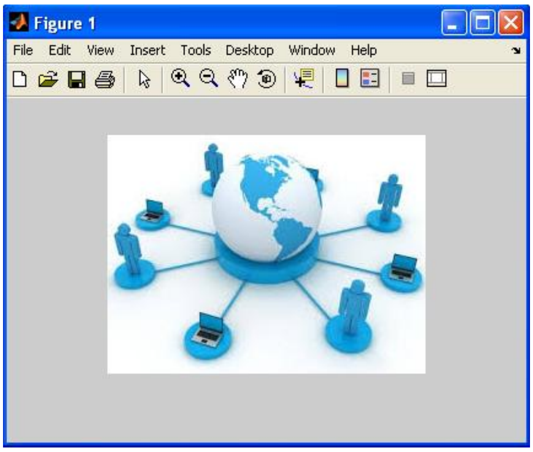
Above mask will find edges in horizontal direction and it is because that zeros column is in horizontal direction. When we will convolve this mask onto an image, it would prominent horizontal edges in the image.

How it Works

This mask will prominent the horizontal edges in an image. It also works on the principle of vertical mask and calculates difference the pixel intensities of a particular edge. As the center row of mask consists of zeros, so it does not include the original values of edges in the image but rather it calculate the difference of above and below pixel intensities of the particular edge. Thus increasing the sudden change of intensities and making the edges more visible.

Implementation of sobel edge detection algorithm

```
a=imread('network.jpg');
imshow(a);
```



Sobel_edge function for detecting edges

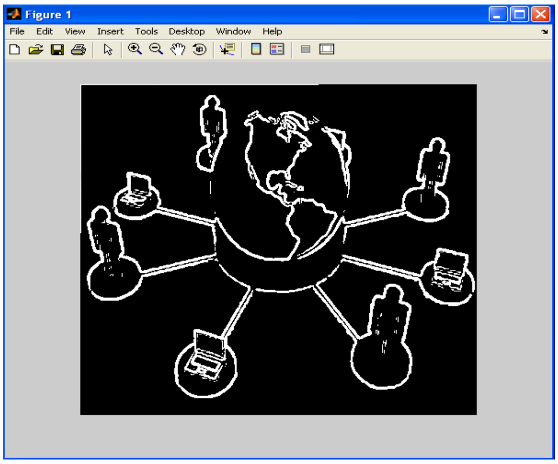
```
function [f]=sobel_edge(img,i,j)
 % sobel kernel
kx=[-1,-2,-1;0,0,0;1,2,1];
```

```
ky=[-1,0,1;-2,0,2;-1,0,1];
sx=0;
sy=0;
fract=0;
for x=-1:1
for y=-1:1
fract=double (img(i+x , j+y))* kx(x+2,y+2);
sx = double(sx+fract);
end;
end;
fract=0;
for x=-1:1
for y=-1:1
fract=double(img(i+x , j+y))* ky(x+2,y+2);
sy=double(sy+fract);
end;
end;
z=uint8(sqrt(sx^2+sy^2))
if(z>=100)
f=255;
else
f=0;
end;
f;
```

Use of above sobel_edge function in our main *sobel.m* file

```
im=imread('network.jpg');
img=rgb2gray(im);
fim=img;
[ht wd ch]=size(img);
im_out=imresize(img,[432 432], 'bilinear');
for i=2:431
    for j=2:431
        fim(i,j)=sobel_edge(im_out,i,j);
    end;
end;
imshow(fim);
```

OUTPUT



Canny Edge Detection Technique

Canny edge detector is one of the most commonly used image processing tools. It detects edges in a very robust manner. Unlike other edge detection operators, canny operation is not very susceptible to noise. It is one of the most important methods to find the edges by separating noise from input image.

It is a much better method because it draws out the features in an image without disturbing its features. The edge detection in this technique is optimized with regard to the following criteria:

- Maximizing the signal-to-noise ratio of the gradient.
- Edge localization for ensuring the accuracy of edge.
- Minimizing multiple responses to a single edge.

Working of Canny Edge Detector

The working of canny edge detection algorithm can be divided into 5 different steps:

1.) Apply Gaussian filter to smooth the image in order to remove the noise.

2.) Find the intensity gradients of the image.

3.) Apply non-maximal suppression to get rid of spurious response to edge detection.

4.) Apply double threshold to determine potential edges.

5.) Edge tracking by hysteresis: Confirm the detection of edges by hiding all the other edges that are not strong and not even connected to any strong edge.

The canny edge detector program has four inputs: input image I, value of smoothing parameter sigma, high threshold Th and low threshold Tl. The horizontal and vertical masks applied in canny edge detection algorithm are based upon the value of sigma. The value of sigma is taken as input and x- and y-derivative masks are generated as output from the value of sigma. In this method, the masks used are 1st derivative of a Gaussian in x- and y-directions. The masks used are also written in two text files for reference.

To generate the mask, the first step is a reasonable computation of the mask size. The mask size should not be too large compared to the lobes of the mask; otherwise it will result in unnecessary computational overhead during convolution. At the same time, the mask size should not be so small to lose the characteristics of primary lobes. We chose mask size based on analyzing the Gaussian and applying a threshold T. After analyzing, the values of mask size obtained by this method for various sigma values are shown below:

Sigma	Size of Mask
0.5	3×3

1	5×5
2	9×9
3	13×13
4	19×19

Scaled masks for sigma=1 are shown below:

Vertical Mask of Canny Operator

15	35	0	-35	-15
69	155	0	-155	-69
114	255	0	-255	-114
69	155	0	-155	-69
15	35	0	-35	-15

Horizontal Mask of Canny Operator

15	69	144	69	15
35	155	255	155	35
0	0	0	0	0
-35	-155	-255	-155	-35
-15	-69	-114	-69	-15

Implementation of Canny edge Detection algorithm

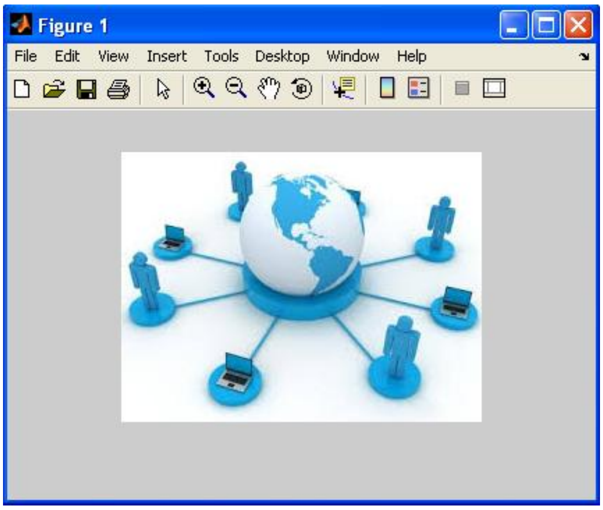
```
function [sFinal,thresh] = cannyop(img, mLow, mHigh, sigma)

if (nargs< 1)
error(' Need a NxNx3 or NxN image matrix');
elseif (nargs ==1)
mLow = 0.5; mHigh = 2.5; sigma = 1;
elseif (nargs == 2)
mHigh = 2.5; sigma = 1;
elseif (nargs == 3)
sigma = 1;
end

origImage = img;

if (ndims(img)==3)
img =double(rgb2gray(img));
end
%CONVOLUTION WITH DERIVATIVE OF GAUSSIAN
dG=dgauss(sigma);
[dummy, filterLen] = size(dG);
```

```
offset = (filterLen-1)/2;
sy = conv2(img, dG ,'same');
sx = conv2(img, dG','same');
[m, n]=size(img);
sx = sx(offset+1:m-offset, offset+1:n-offset);
sy = sy(offset+1:m-offset, offset+1:n-offset);
sNorm = sqrt( sx.^2 + sy.^2 );
sAngle = atan2( sy, sx) * (180.0/pi);
sx(sx==0) = 1e-10;
sSlope = abs(sy ./ sx);
sAorig = sAngle;
y = sAngle< 0;
sAngle = sAngle + 180*y;
binDist =    [-inf 45 90 135 inf];
[dummy, b] = histc(sAngle,binDist);
sDiscreteAngles = b;
[m,n] = size(sDiscreteAngles);
sDiscreteAngles(1,:) = 0;
sDiscreteAngles(end,:)=0;
sDiscreteAngles(:,1) = 0;
sDiscreteAngles(:,end) = 0;
sEdgepoints = zeros(m,n);
sFinal = sEdgepoints;
lowT  =mLow * mean(sNorm(:));
highT = mHigh * lowT;
thresh = [ lowThighT];
gradDir = 1;
indxs = find(sDiscreteAngles == gradDir);
slp = sSlope(indxs);
gDiff1 = slp.*(sNorm(indxs)-sNorm(indxs+m+1)) + (1-slp).*(sNorm(indxs)-sNorm(indxs+1));
gDiff2 = slp.*(sNorm(indxs)-sNorm(indxs-m-1)) + (1-slp).*(sNorm(indxs)-sNorm(indxs-1));
okIndxs = indxs( gDiff1 >=0 & gDiff2 >= 0);
sEdgepoints(okIndxs) = 1;
gradDir = 2;
indxs = find(sDiscreteAngles == gradDir);
invSlp = 1 ./ sSlope(indxs);
gDiff1 =   invSlp.*(sNorm(indxs)-sNorm(indxs+m+1)) + (1-invSlp).*(sNorm(indxs)-sNorm(indxs+m));
gDiff2 =   invSlp.*(sNorm(indxs)-sNorm(indxs-m-1)) + (1-invSlp).*(sNorm(indxs)-sNorm(indxs-m));
okIndxs = indxs( gDiff1 >=0 & gDiff2 >= 0);
sEdgepoints(okIndxs) = 1;
gradDir = 3;
indxs = find(sDiscreteAngles == gradDir);
invSlp = 1 ./ sSlope(indxs);
gDiff1 =   invSlp.*(sNorm(indxs)-sNorm(indxs+m-1)) + (1-invSlp).*(sNorm(indxs)-sNorm(indxs+m));
gDiff2 =   invSlp.*(sNorm(indxs)-sNorm(indxs-m+1)) + (1-invSlp).*(sNorm(indxs)-sNorm(indxs-m));
```

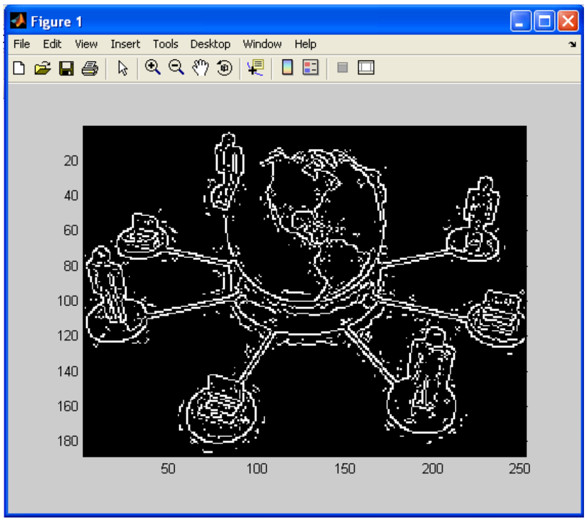
```
okIndxs = indxs( gDiff1 >=0 & gDiff2 >= 0);
sEdgepoints(okIndxs) = 1;
radDir = 4;
indxs = find(sDiscreteAngles == gradDir);
slp = sSlope(indxs);
gDiff1 = slp.*(sNorm(indxs)-sNorm(indxs+m-1)) + (1-slp).*(sNorm(indxs)-sNorm(indxs-1));
gDiff2 = slp.*(sNorm(indxs)-sNorm(indxs-m+1)) + (1-slp).*(sNorm(indxs)-sNorm(indxs+1));
okIndxs = indxs( gDiff1 >=0 & gDiff2 >= 0);
sEdgepoints(okIndxs) = 1;
sEdgepoints = sEdgepoints*0.6;
x = find(sEdgepoints> 0 &sNorm<lowT);
sEdgepoints(x)=0;
x = find(sEdgepoints> 0 &sNorm>= highT);
sEdgepoints(x)=1;
oldx = [];
x = find(sEdgepoints==1);

while (size(oldx,1) ~= size(x,1))
oldx = x;
  v = [x+m+1, x+m, x+m-1, x-1, x-m-1, x-m, x-m+1, x+1];
sEdgepoints(v) = 0.4 + sEdgepoints(v);
  y = find(sEdgepoints==0.4);
sEdgepoints(y) = 0;
  y = find(sEdgepoints>=1);
sEdgepoints(y)=1;
  x = find(sEdgepoints==1);
end

x = find(sEdgepoints==1);
sFinal(x)=1;
figure(1);
imagesc(sFinal); colormap(gray); axis image;
```

OUTPUT

```
x=imread(‘network.jpg’);

Imagesc(x);
```



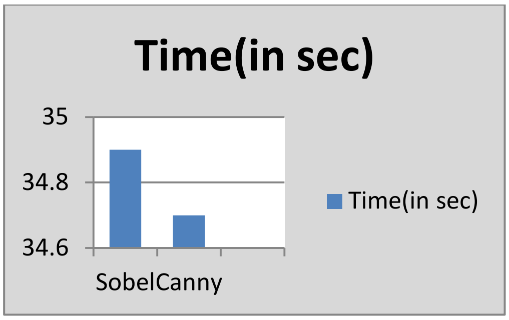
Cannyop(x, .8, .8, .8);



Sobel Edge Detection Technique Versus Canny Edge Detection Technique

1.) **Computation:** The sobel edge detector is simple and less time consuming while canny edge detector is more complex and more time consuming because it involves more computations but still it consumes less time as compared to the sobel edge detection algorithm.

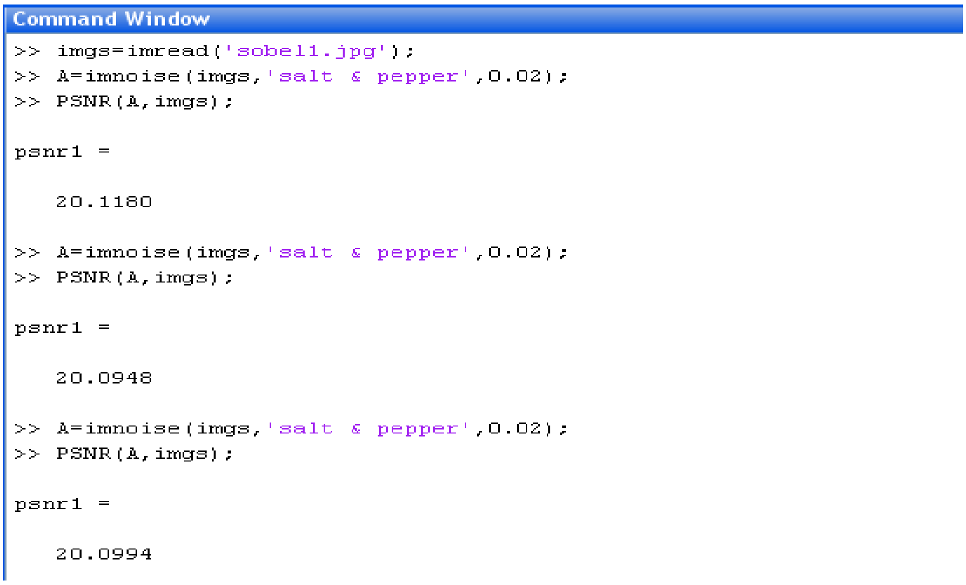
Algorithm	Time(in sec)
Sobel	34.9
Canny	34.7



2.) **Signal-to-Noise Ratio:** As the noise increases, the gradient magnitude of edges also get affected which gives rise to inaccurate edges. The sobel edge detection algorithm has low signal-to-noise ratio. While canny edge detection algorithm has good signal-to-noise ratio. The reason behind this good signal-to-noise ratio is “non-maximal suppression” step involved in canny edge detection algorithm that results in thin edges or single pixel wide edges in the output.

PSNR of Sobel Image:-

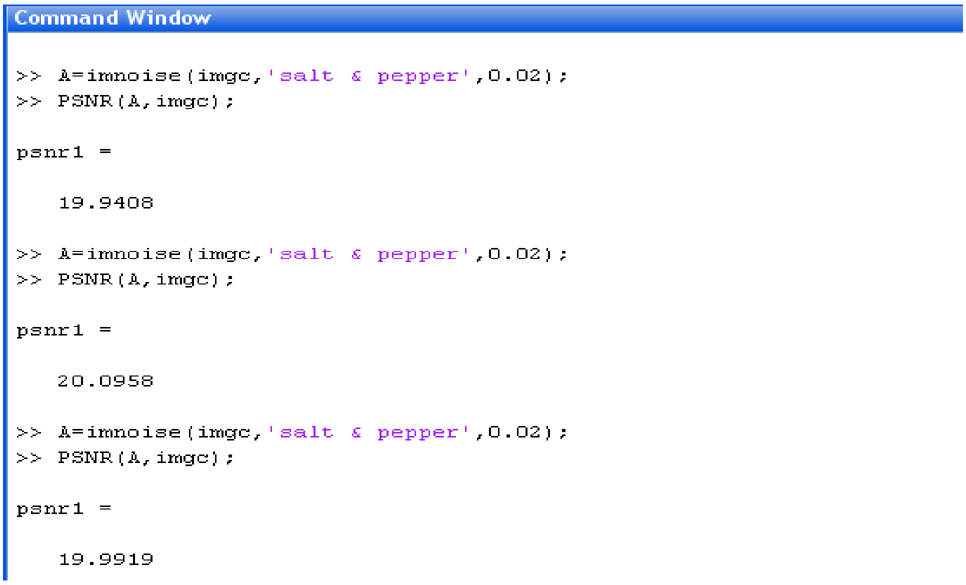
Imgs=imread(‘sobel1.jpg’);



```
>> imgs=imread('sobel1.jpg');
>> A=imnoise(imgs,'salt & pepper',0.02);
>> PSNR(A,imgs);

psnr1 =

   20.1180

>> A=imnoise(imgs,'salt & pepper',0.02);
>> PSNR(A,imgs);

psnr1 =

   20.0948

>> A=imnoise(imgs,'salt & pepper',0.02);
>> PSNR(A,imgs);

psnr1 =

   20.0994
```

PSNR of Canny Image:-

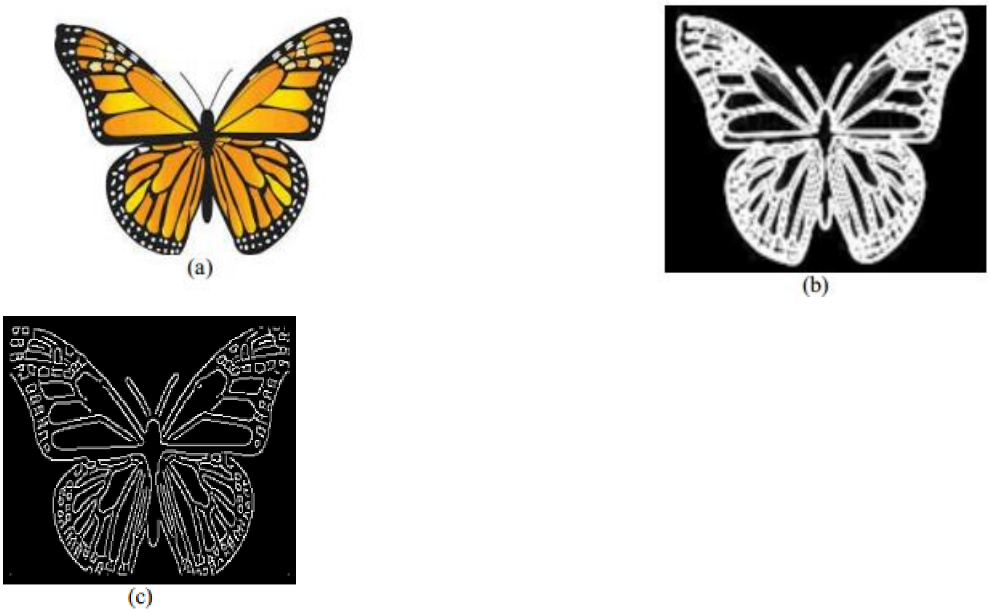
Imgc=imread(‘canny1.jpg’);



```
>> A=imnoise(imgc,'salt & pepper',0.02);
>> PSNR(A,imgc);

psnr1 =

   19.9408

>> A=imnoise(imgc,'salt & pepper',0.02);
>> PSNR(A,imgc);

psnr1 =

   20.0958

>> A=imnoise(imgc,'salt & pepper',0.02);
>> PSNR(A,imgc);

psnr1 =

   19.9919
```

3.) **Texture Based Image:** When we talk about the texture detection in an image, canny edge detection algorithm is much more efficient as compared to the sobel edge detection algorithm.

For Example:-



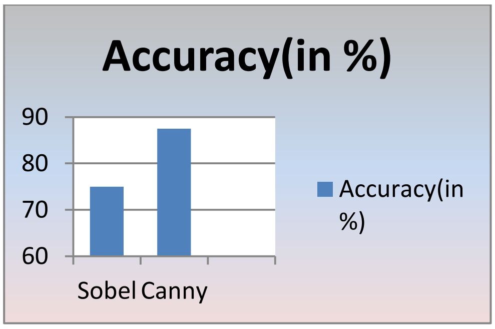
(a) Original Image

(b) Sobel image

(c) Canny image

4.) **Accuracy:** If we are interested in finding the edges of an image containing more number of objects, then canny edge detector is the best method applied for clear and efficient output. Or we can say that sobel edge detector is suitable for simple images while canny edge detector is suitable for both simple as well as complex images.

Algorithm	Accuracy (in %)
Sobel	75%
Canny	87.5%



5.) **Application area:** Sobel edge detection algorithm is mostly used for massive data communication and data transfer while canny edge detection algorithm is highly recommended in medical field for x-ray diagnosis and object recognition.

Future Scope and Conclusion

In this research, we have deeply studied about sobel edge detection technique and canny edge detection technique. From all the study we have done in this research, we realized that canny edge detection algorithm is better than sobel edge detection algorithm in many aspects.

It is less responsive to noise, comfortable with nature, provides good localization, detects sharp edges, produces thin and true edges, and rejects fake and false edges. It acts as the most suitable method for detecting edges till now but a lot of more work can be done over it. For example, canny edge detection algorithm firstly requires the image in gray shade for detecting edges, it cannot directly detect edges from the colored images without converting the image into grade shaded image. So, advanced developments are possible in near future that can detect edges in color image without transforming it into gray image. Or more research for automatic filtration of moving objects in image is under consideration.

Despite of all the advantages and disadvantages of all the edge detection techniques, it is still a very difficult and a very challenging task for the researchers to find the edges without noise that is, a totally noise free image from the original image.

References

1.) The Technology of Night Vision by Harry P. Montoro, ITT Night Vision http://www.photonics.com/EDU/Handbook.aspx?AID=25144

2.) A. Marion An Introduction to image Processing, Chapman and Hall, 1991

3.) Gerhard X. Ritter; Joseph N. Wilson, “Handbook of Computer Vision Algorithms in image Algebra” CRC Press, CRC Press LLC ISBN:0849326362 Pub Date: 05/01/96

4.) http://en.wikipedia.org/wiki/Digital_image_processing

5.) J. Canny, “A Computational Approach to Edge Detection”, IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-8, 6, November 1986, 679-698

6.) G. T. Shrivakshan and Dr. C. Chandrasekar, "A Comparison of various Edge Detection Techniques used in Image Processing", International Journal of Computer Science Issues, Vol. 9, Issue 5, No 1, September 2012.

7.) Raman Maini and Dr. Himanshu Aggarwal, "Study and Comparison of Various Image Edge Detection Techniques", International Journal of Image Processing (IJIP), Volume (3): Issue (1), 2009.

8.) Samta Gupta and Susmita Ghosh Mazumdar, “Sobel Edge Detection Algorithm”, International Journal of Computer Science and Management Research, Volume 2, February 2013.

9.) Muthukrishnan R. and M. Radha “Edge Detection Techniques for image Segmentation” International Journal of Computer Science & Information Technology (IJCSIT) Vol3, No 6, Dec 2011