Project Titled

"Exploring Methods to Improve Edge Detection with Canny Algorithm"

Submitted by

Aman Mehta(111080050)
Harshal Lehri(111080953)
Prasad Thakur(111080046)
Sayali Tambe(111081021)

B.tech(Information Technology)

2014-15

Under the Guidance

Prof. V. K. Sambhe

Department of Computer Science and Information Technology.

Veermata Jijabai Technological Institute

(An Autonomous Institute affliated to Mumbai University)

Mumbai-400019

STATEMENT OF CANDIDATE

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and references the original sources. We also declare that we have adhere to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any data in our submission. We understand that any violation of the above will be cause for disciplinary action by Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Aman Mehta
111080050

Harshal Lehri
111080953

Prasad Thakur
111080046

Sayali Tambe
111081021

Contents

1 Introduction **1**
- 1.1 Image as a Signal . . . 2
- 1.2 Theory of Edge Detection . . . 4
  - 1.2.1 Edge Definition . . . 4
  - 1.2.2 The Biology behind Edge Detection . . . 4
  - 1.2.3 Fundamental Steps in Edge Detection . . . 6
  - 1.2.4 Evaluation of Detectors . . . 7
- 1.3 Edge Detector Types . . . 8
  - 1.3.1 Convolution . . . 8
  - 1.3.2 The Gaussian Filter . . . 9
  - 1.3.3 Gradient Based Edge Detection . . . 11
  - 1.3.4 Second Ordered Derivative Based Edge Detection . . . 13
  - 1.3.5 The Canny Edge Detector . . . 15
- 1.4 Problems with Standard Edge Detection Method . . . 17

2 Literature Survey **22**

3 Methods Explored **25**
- 3.1 Small Scale Edge Detection using Recursion . . . 26
  - 3.1.1 Basics of Recursion . . . 27
  - 3.1.2 Using Recursion with Canny Algorithm . . . 28
- 3.2 Edge Filtering by determining Conditional Probability of a point being an Edge . . . 29
  - 3.2.1 Polynomial Regression . . . 30
  - 3.2.2 Edge Linking . . . 30
  - 3.2.3 Algorithm for the Edge Filtering Technique . . . 32
- 3.3 Results . . . 32

4 Analysis of the Results **38**

5 Future Work **46**

6 Conclusion **48**

List of Figures

1.1 Profiles and types of edges 2
1.2 3D view of a Image as a signal 3
1.3 Working of Ganglion cells 5
1.4 Convolution on an image 9
1.5 Process of convolution for 2D discrete images 10
1.6 Outputs of Popular Edge Detectors 18
1.7 Edge Detection on images with different context 20
1.8 Effect of placement of object on Edge Detection 20

3.1 Original image of vegetables(cluster of objects) 25
3.2 Canny Edge of a cluster of objects 25
3.3 Effect of Size of image on Edge Detection 26
3.4 Flow of Recursion on a Image 28
3.5 Probability generated for Block 1 30
3.6 Probability generated for Block 2 30
3.7 Probability generated for Block 3 30
3.8 Probability generated for Block 4 30
3.9 Probability of different blocks 30
3.10 Result for the Penguin Image 33
3.11 Result for the Lena Image 34
3.12 Result for the Photographer Image 35
3.13 Result for the Desert Image 36
3.14 Result for the Vegetable image 37

4.1 Comparing Output of Penguin Image and Recursive Image . 39
4.2 Effect of size of kernel on the recursive algorithm 40
4.3 Effect of image size on the recursive algorithm 40
4.4 Comparing Output of Canny with the refinement algorithm . 41
4.5 Effect of length of threshold on the refinement algorithm . . . 42
4.6 Effect of probability threshold on the output of refinement algorithm 43
4.7 Effect of Canny truth value on the refinement algorithm . . . 44
4.8 Effect of neighborhood size on the refinement algorithm . . . 45

Abstract

Edge Detection is an important step in any image processing algorithm, and the quality of edges detected determines the performance of the successive steps performed. The objective of this report is to analyze and design algorithms that when used in conjunction to the Canny Edge Detector, gives better edge detection performance and minimizes influence of external noise which leads to detection of false edges. All the algorithms have been designed using OpenCV libraries. We analyze two areas in Edge detection 1. Applying Canny at a smaller scale and 2. Post processing the results from the Canny algorithm and filtering unwanted edges. The results improve the output of Canny, though unwanted edges are still detected and shown in the final results. Hence the algorithms hold potential but are far from giving any significant improvement in their current state and future work will be done to explore further improvements.

Chapter 1

Introduction

In computer vision, Edge detection is a process to find the set of pixels that represent the boundary of an disjoint regions in an image. It tries to capture the properties of an image such as the discontinuities in the geometrical and physical characteristics of an object. The most common discontinuities in an image are step edges, line edges and junctions where the 2D features form meet. Edges characterize the boundary of objects in an image. Edge detection is important for image segmentation, since many image processing algorithms require first to identify the objects and then process them. Edge detection allows users to observe features of an image showing an abrupt change in gray level or texture indicating a boundary between two different regions in an image. It finds a lot of practical applications in medical imaging, face detection, fingerprint verification, traffic control system and many more. The innumerable applications of edge detection has made it one of the most researched topics in image processing. Several edge detection algorithms have been developed for extracting edges from an image. Over the years, there have been several operators designed for this purpose but provide varying results and hence often are application specific rather than a generalized algorithm. The purpose of edge detection is to localize the variations in an image and must be efficient and reliable to reduce the error in the subsequent processing stages of the application.

However edge detection is not without its challenges. The image on with the detection is carried out has noise, due to random variation of brightness or color information in images due to electronic noise. There are several forms of noise like Gaussian noise, salt pepper noise and shot noise to name a few. With noise the edge detection algorithm is more likely to detect false edges which degrades the quality of edges detected. A good edge detector increases the probability of of marking an actual edge pixel while decreasing the probability of marking a false edge.

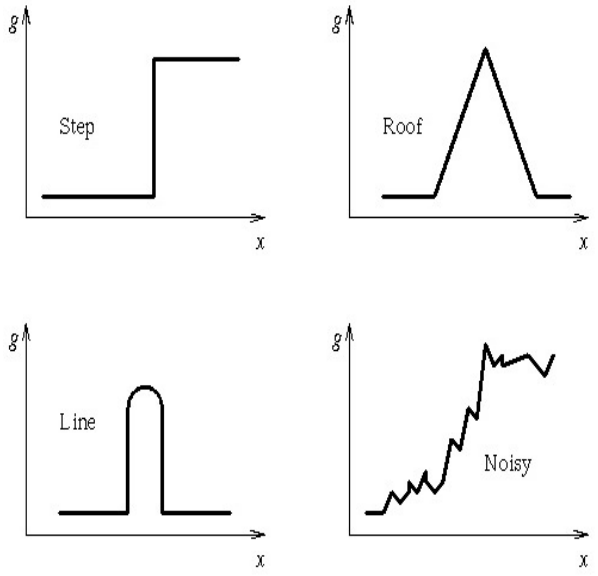


Figure 1.1: Different profiles of an edge.

The report explains the basics of edges in an image and the mathematics behind them. We then explore the popular edge detectors used along with how they work. We then discuss the variety of approaches to process the edge image with the aim to better results produced by the standard edge detection algorithm, particularly the Canny operator. The Canny operator is particularly popular and widely used algorithm. We try to improve its results by eliminating false edges detected by it. We analyze the results produced by our algorithms and finally conclude with the future prospects in this area of computer vision.

1.1 Image as a Signal

Mathematically a signal is a function of one or more independent variables. An image can be considered a function of three variables(spatial coordinates x and y and time). However images are usually constant over time(unlike videos) and hence are described as a function of x and y itself. The x and y represent the spatial coordinates with each position in an image and the function returns the grey value at that location. Images can also be multi channel ie each position x,y corresponds to three values(RGB). For an image I(x,y) the energy is given by:

$$E = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |I(x,y)|^2 dxdy \tag{1.1}$$

Images being signals can exhibit periodicity. A 1D signal is said to be peri-

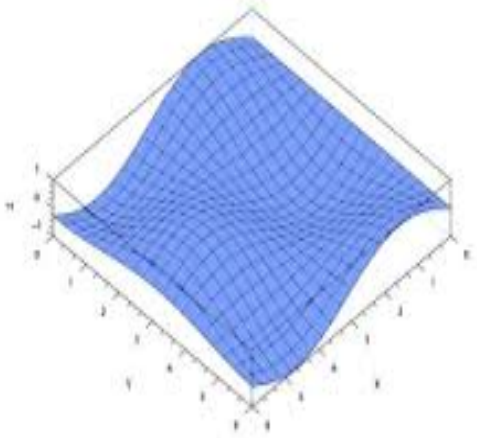


Figure 1.2: 3D representation of a signal z = f(x,y). An image is a 2D signal with each value of z depicting the gray value or color at spatial positions x and y.

odic with a time period T iff $g(t+T) = g(t)\ \forall$ t ϵ R. A 2D image is a periodic signal with a period (T_x,T_y) iff $g(x+T_x, y) = g(x, y+T_y) = g(x, y)\ \forall x, y\ \epsilon$ R.

A lot of operations are applicable on such spatially periodic signals. A common operator is the Fourier transform to convert it into different set of coordinates u and v. u and v are said to be the spatial frequencies. The 2D Fourier transform can be written as:

$$G(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(x, y) e^{-i2\pi(ux+vy)} dxdy \tag{1.2}$$

The inverse 2D Fourier Transform can be written as:

$$I(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} G(u, v) e^{i2\pi(ux+vy)} dxdy \tag{1.3}$$

Other transforms like the Laplace transform, Hartley transform and Radon Transform can also be extended to 2D signals.

Once an image is captured, the image must be treated as a discrete signal rather than a continuous functions since the image can be recorded on a 2D grid. The conversion is done by a process called sampling. The most common form of sampling is uniform rectangular sampling which is performed by using small sampling intervals Δ_x and Δ_y such that if I(x,y) represents the continuous image signal and $I_d(x, y)$ represents the digitized image, $I_d(n, m)$ $= I(n\Delta_x, m\Delta_y)$. In general one cannot recover the continuous signal back

from the discrete samples. However for band limited functions, we can apply Nyquist-Shannon theorem to find the sampling rate. Hence if in the spatial frequency u-v domain the signal is limited by u_{max} and v_{max}, and if $\Delta_x < 1/2u_{max}$ and $\Delta_y < 1/2v_{max}$ then we say the image is adequately sampled, otherwise we say the image is undersampled. The image can be reconstructed using the sinc function given by $sinc(x) = sinc(x)/x$. The figure of a 2D image signal is shown in Fig 1.1.

1.2 Theory of Edge Detection

1.2.1 Edge Definition

Edges are points where there is a boundary between two image regions. It can be of any arbitrary shape. They can represented as a set of points that show strong gradient or large transition in surrounding intensity. The edges with similar gradient orientation are grouped together. They are one dimensional structures. Physical edges correspond to significant variation in reflectance, illumination, orientation and depth of the scene surfaces. Physical edges can be thought of as changes in the image intensity function. As mentioned there are variations of physical edges like step edges, line edges and junction edges. Steps are the most common type of edges encountered, and represent a sudden change in intensity due to different phenomenon like shadows on a surface. They are maxima or minima points in the image. A step edge is usually a combination several single steps. Line edges result from mutual illumination between objects that are in contact or from thin objects present in the background. A junction edge is a point where two or more edges meet. It is a 2D feature and will have a sharp point where all the lines meet. The junction can be modelled in variety of ways: T,L,Y and X. They are points of high curvature.

Though a popular definition of an edge can be thought of as the extremum of the first derivative in one or more directions. Due to noise this definition is not particularly useful and depends on direction in which it is applied. Hence a more useful definition of an edge can be thought of as points with zero crossing when the Laplacian operator(often combined with the Gaussian smoothing operator to form the popular Laplacian of Gaussian or the LoG operator) is applied on the image. This operator is isotropic i.e similar in all directions.

1.2.2 The Biology behind Edge Detection

Marr and Hildreth's Theory of Edge Detection explains the purpose and process of edge detection quite well[4]. Edge Detection was inspired from research in visual systems of higher organism. The physiological process was

focused more, which included alignment of cells in the visual cortex of the retina. Rather than acting as a camera, where the elements we see are copied as they are in the brain, the eye detects specific visual features like sudden intensity changes. The eye also acts as a natural filter, removing unwanted high frequency noise and focusing on the lower frequencies containing more information. H.B.Barlow's tests on the retina of a frog provided the first evidence of cells specifically evolved to detect changes in intensity. The cells were like "on-off" switches that detected both light to dark and dark to light transitions.

There were two popular notions of the function of early information processing in visual systems of complex organisms by the 1970's. Hubel and Wiesel theorized that the visual cortex consists of bar and edge shaped receptive fields that act as natural edge detectors and feature extractors. Campbell and Robson's experiments however showed that the cells processed visual information parallel-ly and were not tuned to specific orientation rather acted as a kind of spatial Fourier Analyzer. The main point against the linear feature detector idea is that cells tuned to detect a particular bar or edge would respond to other stimuli as well and was not specific enough that it could be called a feature detector. Similarly the interpretation of cortex cells being spatial Fourier analyzers has received criticism since the bandwidth of the channels is not narrow enough and receptive fields have a definite spacial localization rather than being independent of orientation.

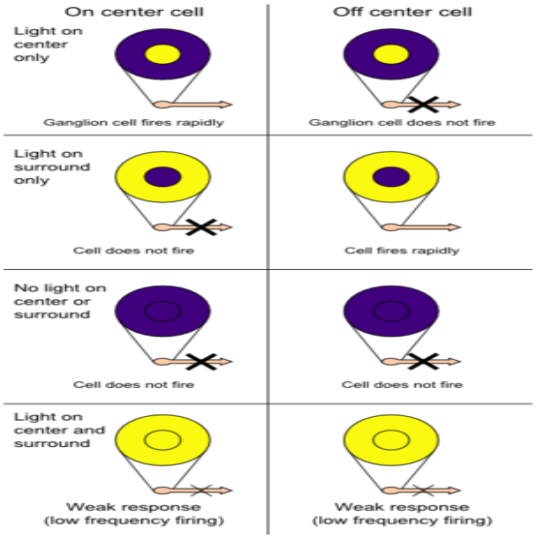


Figure 1.3: Internal working of the ganglion cells to different intensities of light.

1.2.3 Fundamental Steps in Edge Detection

The input to any edge detector is a discrete, digitized image and produces an edge map. The edge map is usually a binary image with the edge points represented as 1's(light region) while the rest of the image is represented as 0's(dark regions). There are two classes of edge detectors. The first includes detectors which do not use a priori knowledge about the scene and the edge to be detected. These are not influenced by other components of computer vision system nor by contextual information. They are based on local processing by analyzing the surrounding or neighborhood of the pixels. The second class of detectors are contextual i.e are application specific and are guided by the results of other components of the system.

The most common steps for any edge detectors includes three steps: image smoothing, image differentiation and edge labeling. Smoothing reduces the noise in the image, differentiation finds the points of interest(maxima-minima or points of zero crossing) in an image. Labeling identifies the true edges from the false ones(generated due to noise) and increases the signal-to-noise ratio of the image by suppressing the false edges. All the three steps are interlinked. The performance of one directly influences the performance of the next stage. The amount of noise reduced by smoothing determines the number of false edges detected by the differentiation operation which affects the performance of the edge labeling step. This specification is still not complete and scale and type of edge detector must also be stated.

Smoothing while reducing noise also causes information loss. The optimal edge detectors hence must find the best compromise between the two. The search of an optimal filter can be explained by Regularization theory. Regularization theory tries to solve an ill-poised problem(a mathematical model with no guarantee that a solution exists,the solution is unique and stable) by introducing additional information in order to solve it. Poggio and Torre show that the differentiation can be regularized by convolution of the image with derivatives. The linearity of the filter, duration of the impulse response and its invariance to rotation all determine its performance.

The Differentiation operator is characterized by its order, its invariance to rotation and its linearity. An operator $O_{x,y}$ is invariant to rotation iff $O_{x,y}$=$RO_{x,y}$ where R is the rotation matrix. $O_{x,y}$ is linear iff for all positive scalars α and β and for all f(x,y) and g(x,y), we have $O_{x,y}(\alpha f(x,y) + \beta g(x,y))$=$\alpha O_{x,y}(f(x,y))+\beta O_{x,y}(g(x,y))$. The first order gradient is defined as $\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$. The gradient direction is perpendicular to the edge orientation. The gradient modulus operator is non-linear and invarient to rotation. By steering's theorem derivatives of the image in any direction can be expressed as the weighted sum of the derivatives of the image in particular

directions. Hence the gradient modulus is often calculated using one of two other metrics:$\left|\frac{\partial}{\partial x}\right| + \left|\frac{\partial}{\partial y}\right|$ and $max(\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$. This is more efficient however the accuracy is reduced. The second order derivative can be defined using Laplacian and the second-order directional derivative along the gradient direction. The operators are defined by:

$$\nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad and \quad \frac{\partial^2}{\partial \vec{n}^2} = \frac{\partial^2}{\partial x^2}\cos^2(\psi) + \frac{\partial^2}{\partial x \partial y}\sin(\psi)\cos(\psi) + \frac{\partial^2}{\partial y^2}\sin^2(\psi) \tag{1.4}$$

$\vec{n}$ is the gradient direction. The Laplacian operator is linear and rotationally symmetric, whereas the second directional directive is neither linear nor invarient to rotation. The appropriate sequence of these operators depends on the linearity of the differentiation operator. For a linear smoothing operator, convolution is association and commutative and hence it the order in which smoothing and differentiation is applied is immaterial. For non-linear filters, smoothing must precede differentiation operation.

Edge labeling involves localizing edges and increasing signal-to-noise ratio by suppressing false edges. The simplest method is thresholding the gradient modulus. A better edge thresholding method involves non-maximal suppression, wherein the local maxima along the direction of the gradient vector is maintained, while the others are suppressed. For second order detectors the localization of zero-crossings, the output of the detector at a pixel is compared with the neighboring pixels to the left and below it. If the three points do not have same signs, there is a zero-crossing. The edge labeling involves thresholding the edges to determine if an edge detected is a true edge or a false edge. The threshold is a function of edge characteristics, properties of the smoothing filter, and properties of the differentiation operator.

1.2.4 Evaluation of Detectors

Evaluation process of an edge detector requires criteria or reference describing the characteristics of the edge to be detected. The evaluation can be subjective as well as objective. Subjective evaluation involves judging the results by human subjects who rate the detector. The evaluation is rough since it is difficult for humans to distinguish between two close grey levels or two close orientation. Objective evaluation measures the performance of an edge detector. Abdou and Pratt have proposed a measure, called the figure of merit which is a combination of three factors: non-detection of three edges, detection of the false edges and edge delocalization error. However it is difficult to point out the type of error committed by the detector. There are four major types of errors, given by Venkatesh and Kitchen: non-detection of true-edges, detection of false edges, detection of several edges

instead of an edge one pixel wide, and edge delocalization error.

Subjective and objective evaluations can be used together to evaluate edge detectors. This is based on statistical analysis. One method is suggested by Heath. For each edge detector a combination of parameters is chosen and the resulting edges are presented to eight judges. They rate the detector on a scale of 1 to 7. A rating of 1 means the edge cannot be coherently organized into an object while 7 means that all the edges are relevant for recognizing an object. From this experiment the ratings are analyzed statistically and the best parameters are selected for each edge detector and for each image. The second experiment concerns comparison between edge detectors and the original images and corresponding edges are presented to sixteen judges for rating.

1.3 Edge Detector Types

Many Edge detection methods have been developed for extracting edges from a digital image. The can be broadly classified into two categories: Gradient based and Laplacian based. The gradient based approach detects edges by looking for the maxima or minima of the first derivate of the image. The Laplacian method on the other hand tries to detect edges by searching for the zero crossing of the second derivate of the image. However before looking into the various type of edge detectors we will discuss an important operation used during edge detection called convolution.

1.3.1 Convolution

In functional analysis, convolution is a mathematical operator on two function f and g to produce a third function h that is a modified version of one of the original functions, given the area overlap between the two functions as a function of the amount that one of the original functions is translated. For one dimension it can be written as:

$$h(x) = (f * g)(x) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau \qquad (1.5)$$

If functions are seen as signals, the convolution operator on I(x,y) and G(x,y) produces a new signal H(x,y) such that:

$$H(x,y) = (f * g)(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\alpha,\beta)g(x-\alpha,y-\beta)d\alpha d\beta \qquad (1.6)$$

Convolution is commutative, associative, distributive and invariant to scaling. The interesting property about convolution is its property under differ-

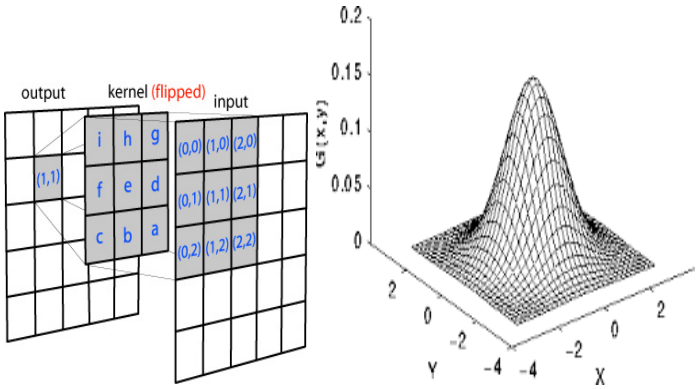


Figure 1.4: (a)Process of convolution for 2D discrete images (b)A Gaussian Filter in 2D

entiation given by

$$\frac{\partial}{\partial x_i}(f * g) = \frac{\partial f}{\partial x_i} * g = f * \frac{\partial g}{\partial x_i} \tag{1.7}$$

This is under the assumption that both f and g are differentiable under x_i. These properties make convolution a very important operator in image processing. Consider that the function f is the signal image while the function g is the filter image that will detect edges. The resulting image is the edge image. The simplest idea of such an image should be a filter that returns one during drop in intensity, zero otherwise. However the choice of filter is difficult and many times does not exist. However Canny proved that the best filter is the Gaussian filter under approximation. For 2D discrete images we approximate the convolution operator as:

$$h(x, y) = (f * g)(x, y) = \sum_{\alpha=0}^{\alpha=n} \sum_{\beta=0}^{\beta=m} f(x, y)g(x - \alpha, y - \beta)$$

Here n and m are the limits of the kernel.

1.3.2 The Gaussian Filter

From the above discussion we saw that taking the convolution of the image smoothens it out. We consider a special filter called the Gaussian Filter. A Gaussian Filter is a filter whose impulse response is a Gaussian function. The smoothen out the shot function but do not change its properties drastically. Hence it is pretty useful to remove random noise while keeping original edges in the image. The Gaussian function can be given by:

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

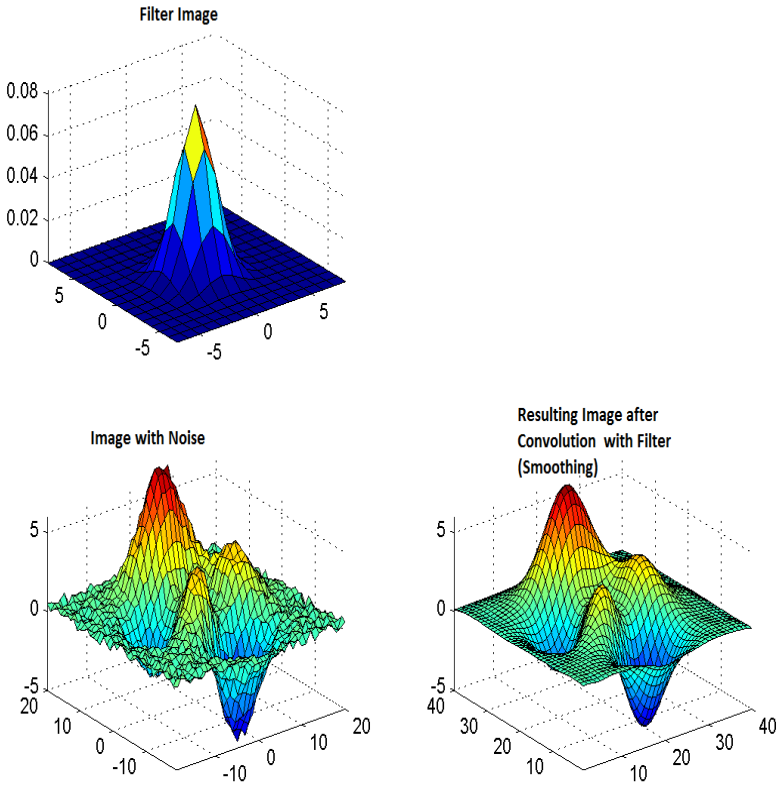


Figure 1.5: Process of convolution for 2D discrete images

The idea of Gaussian smoothing is to use this 2-D distribution as a 'point-spread' function, and this is achieved by convolution. Since the image is stored as a collection of discrete pixels we need to produce a discrete approximation to the Gaussian function before we can perform the convolution. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point. One such kernel is shown:

$$G = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 416 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Note that the function is isotropic in 2-D ie similar in all directions. The convolution with this filter is also faster since the Gaussian can be split into its X and Y components which can be independently applied. This is the reason why the Gaussian filter is popular.

The σ defines the spread of the function. As the spread increases more pixels are involved during convolution and hence increases the false edges detected(noise). Hence smaller Gaussian filter should be used.

1.3.3 Gradient Based Edge Detection

Consider an image as a function of two variables i and j such the I(i,j) returns the intensity value of the image at point (i,j). The point (i,j) is the pixel of the image. Depending on the image a I(i,j) can be a single value (Gray scale images) or be multivalued(like RGB and other multichannel images). Hence for the function I(i,j), the derivative of the function can be written as:

$$\nabla I = \hat{i}\frac{\partial I(i,j)}{\partial i} + \hat{j}\frac{\partial I(i,j)}{\partial j}$$

Where :

$\frac{\partial I(i,j)}{\partial i}$ is the gradient in the horizontal direction

$\frac{\partial I(i,j)}{\partial j}$ is the gradient in the vertical direction

The gradient magnitude is computed as:

$$|G| = \sqrt{\left(\frac{\partial I(i,j)}{\partial i}\right)^2 + \left(\frac{\partial I(i,j)}{\partial j}\right)^2}$$

and the gradient direction is computed as

$$\theta = \arctan\left(\frac{\partial I(i,j)}{\partial j} / \frac{\partial I(i,j)}{\partial i}\right)$$

However the image is not a continuous function, rather a discrete one. Hence we must define the derivative of a discrete function. There are several classical operators that do so which we will describe below.

Roberts Operator

The Roberts operator performs a simple and quick to compute 2-D spatial gradient measurement on an image. It thus highlights regions of high spatial

gradient which often correspond to edges. This gradient based operator is computed by convolution of the image using two 2 x 2 matrices. The two kernels are given as:

$$D_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad D_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

The magnitude of the gradient component can be computed as $|D|=\sqrt{D_x^2 + D_y^2}$ however is often approximated as $|D|=|D_x| + |D_y|$. The orientation is given by: $\arctan(\frac{D_y}{D_x})$. The main reason for using the Roberts Cross operator is that it is very quick to compute. Only four input pixels need to be examined to determine the value of each output pixel, and only subtractions and additions are used in the calculation. In addition there are no parameters to set. Its main disadvantages are that since it uses such a small kernel, it is very sensitive to noise. It also produces very weak responses to genuine edges unless they are very sharp.

Sobel Operator

Sobel operator is used to compute an approximation of the gradient of the image intensity function of the edge detection. At each pixel of an image the Sobel operator gives either the corresponding gradient vector or the normal vector. It uses the following 3 x 3 kernels:

$$D_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad D_y = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

These masks are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one mask for each of the two perpendicular orientations. The masks can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these D_x and D_y). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude and orientation are computed similar to Roberts operators.

Prewitt Operator

The Prewitt Edge filter is use to detect edges based applying a horizontal and verticle filter in sequence. Both filters are applied to the image and summed to form the final result. At each point in the image, the result of the Prewitt operator is either the corresponding gradient vector or the norm

of this vector. The Prewitt operator is similar to a sobel operator but has a different kernel and gives better performance the Sobel. The kernel used is given below:

$$D_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \qquad D_y = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Algorithm for Detecting Edges using Gradient Operators

The algorithm for identifying if a pixel is an edge point or not is done using the simple algorithm given below:

1. Preprocess the image using filters to reduce the effect of noise
2. Convolve the image in the horizontal direction using the x-Mask of the gradient operator (Sobel, Roberts, Prewitt etc)
3. Convolve the image in the vertical direction using the y-Mask of the gradient operator
4. Determine the magnitude of the gradient for each pixel and compare it to a threshold
   (a) If the magnitude is greater than the threshold, mark the pixel as an edge image
   (b) Else the pixel is not an edge image
5. Repeat step 4 for all pixels in the image

Note however that the algorithm does not make use of the gradient direction operator.

1.3.4 Second Ordered Derivative Based Edge Detection

In this method of Gradient Detection we compute the second order derivative of the function I(i,j) and we mark a pixel as an edge point if the second order derivative at that point is zero. The generalized formulae for this approach is given below:

$$\nabla^2 I = \frac{\partial^2 I(i,j)}{\partial^2 i} + \frac{\partial^2 I(i,j)}{\partial^2 j}$$

A directional vector can also be directional operator for second order differentiation, given as:

$$\frac{\partial^2}{\partial \vec{n}^2} = \frac{\partial^2}{\partial x^2}\cos^2(\psi) + \frac{\partial^2}{\partial x \partial y}\sin(\psi)\cos(\psi) + \frac{\partial^2}{\partial y^2}\sin^2(\psi)$$

Marr Hildrith Operator

The Marr-Hildrith Operator was a very popular edge detector before Canny proposed his algorithm. The difficulty with processing natural images is that the changes occur over a wide range of scales. No single filter can optimally reduce noise at all scales. The basic idea is then to take local averages of the image at various resolutions and then detect the changes in intensity that occur at each of the resolutions. We need to hence determine the optimal smoothing filter and detect intensity changes. Filtering is necessary to reduce the range of scales over which the intensity changes take place. The second thing we need to consider is the physical phenomenons that give rise to intensity changes like illumination and surface reflectance are spatially localized. The two requirements however lie in different domains, the first one being in the frequency domain and the second in the spatial domain. By the uncertainty principle we cannot achieve both at the same time. There is only one distribution that optimizes this relationship which is the Gaussian distribution given by:

$$G(x) = \frac{1}{\sigma(2\pi)^{1/2}} e^{\frac{-x^2}{2\sigma^2}} \tag{1.8}$$

Intensity changes can be detected by either finding the maxima or minima of the first derivative or finding zero crossings in the second derivative of the image. Hence we can write it as:

$$f(x,y) = D^2(G(x,y) * I(x,y)) \tag{1.9}$$

where G(x,y) is the Gaussian Kernel, I(x,y) is the image and the * is the convolution operator. D^2 represents the second order directional derivative. $D^2 * G$ is the mexican hat operator. It is important to note that the direction must be independent of orientation and hence Marr-Hildret suggested using the Laplacian operator ∇^2.

It used both the Gaussian operator to reduce noise and the Laplacian Operator to calculate the second order derivative of the image. For discrete images, it is necessary to approximate the differentiation stage. The second order derivative(Laplacian) of the discrete image can be approximated as:

$$\frac{\partial^2 I(i,j)}{\partial^2 i} = I(i,j+1) - 2I(i,j) + I(i,j-1)$$

$$\frac{\partial^2 I(i,j)}{\partial^2 j} = I(i+1,j) - 2I(i,j) + I(i-1,j)$$

The Laplacian can be implemented using a kernel of size 3x3:

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

The good thing about the Marr Hildrith Operator is that it is an isotropic in nature and utilized only one mask, hence cheaper to implement. Also since it differentiates twice it is more sensitive to noise. Since a Gaussian Filter is applied to the image, we can combine the two operations to a single Laplacian of Gaussian matrix (LoG). A 5x5 example of the matrix is given below:

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

1.3.5 The Canny Edge Detector

The Canny edge detector[1] is one of the most sophisticated and popular edge detectors available. It is a gradient based edge detector, though it offers much more sensitivity to noise. Canny consider three performance criteria and accordingly tries to optimize them. The three performance which are given as follows:

1. Good detection: The probability of failing to mark a real edge should be low and the probability of marking a false edge should also be low. This corresponds to maximizing the signal-to-noise ratio.

2. Good localization: The points marked as edge points should be as close as possible to the center of the true edge.

3. Only one response to a single edge: This is implicitly captured in the first criterion since when there are two responses to the same edge, one of them must be considered false. This is achieved by non-maximal suppression.

The first two criterions are related. The Signal to Noise ratio is first found. For a given edge G(x) and filter response f(x), we can find the signal response function as:

$$H_G = \int_{-W}^{W} G(-x)f(x)dx$$

. The fine impulse response is bounded by [-W,W]. The rms value of the noise is given by

$$H_G = n_0\sqrt{\int_{-W}^{W} G(-x)f^2(x)dx}$$

. The n_0^2 is the mean square noise amplitude per unit length. Hence the Signal-to-Noise ratio is given by:

$$SNR = \frac{\left|\int_{-W}^{W} G(-x)f(x)dx\right|}{n_0\sqrt{\int_{-W}^{W} G(-x)f^2(x)dx}} \tag{1.10}$$

To quantify the localization, we get the local maximum at point x=x_0 if

$$H_n^{'}(x_0) + H_G^{'}(x_0) = 0$$

where point x_0 is point of local maxima. We can approximate this equation to

$$H_G^{''}(0)x_0 \approx -H_n^{'}(x_0)$$

. Finding the expected value of x_0 we find the expectation of the above equation which is given by:

$$E[x_0^2] \approx \frac{n_0^2\int_{-W}^{W} f^{'2}(x)dx}{\left[\int_{-W}^{W} G^{'}(x)f^{'}(x)dx\right]^2} = \delta x_0^2$$

. He uses the reciprocal of δx_0 for quantifying localization. Hence we try to maximize the product of the two criterions: Localization and SNR.

The third criterion involves finding a single maxima. Due to the nature of noise having multiple step responses, there are multiple local maxima in the vicinity of the actual maxima. Hence Canny derives that the number of maximas in a window W is given by $N_n = \frac{2W}{kW} = \frac{2}{k}$. Here k is the fraction of the maximum distance between two maximas to the window size W. Fixing k fixes the number of noise maximas that could lead to a false edge. Through calculus of variations Canny found out that the optimal response function f(x) can be approximated to the derivative of Gaussian. Due to properties of Gaussian function, the image is first smoothened with Gaussian filter and then the derivative is applied.

The algorithm for the canny algorithm is given below:

1. Apply noise reduction using Gaussian Filter on the image. This will make the image smooth and hence offer better edge detection

2. The next step is finding the gradients of the image. This is done by finding the first order derivatives of the Image using operators like Sobel operator. This can be achieved by convolving the kernel over the image pixel by pixel.

3. The algorithm then carries out non maximal suppression. This is done as follows: For every pixel (i,j)

(a) Round the gradient of (i,j) to the nearest multiple of 45^0. Now compare the pixel to the neighboring pixels in the direction of the gradient.

(b) If the pixel is the strongest(highest intensity) in the direction of the gradient, then keep the gradient else remove it.

4. The final step is hysteresis thresholding. For a pixel (i,j) having gradient G:

   (a) IF G $< t_{low}$ then discard the edge point

   (b) IF G $> t_{high}$ then keep the edge

   (c) Otherwise IF t_{low} < G < t_{high} and any of its neighbors in the 3x3 region around it have gradient greater than t_{high}, keep the edge

   (d) IF such a neighboring pixel cannot be found in the 3x3 region, search the 5x5 region. If a pixel is found with G $> t_{high}$, then keep the edge

   (e) Else discard the edge

Canny offer optimal edge detection due to the following reasons:

1. It is less sensitive to noise, since the application of the Gaussian filter smoothens out the image

2. It removes "streaking". Streaking means that if an edge is just below the set threshold it will remove it leaving disconnected final edges. However since Canny uses two thresholds $t_{high} and t_{low}$, the edge detector can connect the discontinuities by observing the edges in the lower threshold image.

3. Canny offers good localization of edges and utilizes gradient of the edge to generate thin one pixel wide edges which other algorithms do not offer.

1.4 Problems with Standard Edge Detection Method

Edge detection is an non-trivial task. Edge detection strongly relies on the intensity difference between two neighboring pixels. It largely depends on the illumination when the image is taken. All edge detection have to apply some filter to reduce noise. Unfortunately this also spreads the edges in the image thereby making it more difficult to detect edges in it. There are several external factors like random noise and illumination that affects the performance of edge detection algorithms. Edge detection also involves several different issues, like edge linking that may lead to detection of false

Figure 1.6: Left to right:(a)Original Image (b) Output of Canny Edge Detector (c) Output of Sobel operator (d) Output of Laplacian of Gaussian Operator

edges. It is also very difficult to generalize the method of thresholding to be applied on the image.

Canny offers a very adaptive and optimal solution to edge detection, which can be inferred from its popularity. The algorithm is sensitive in noise and adaptive in nature. It also detects sharper and thinner edges. Due to its optimal nature, it is used as a benchmark over which several research projects are carried to improve the method. Canny has been customized to suit the applications it is used in, for example analyzing MRI images and in intelligent traffic systems. Thus it is worth while to explore methods that aid Canny for better edge detection.

Like any other image processing algorithm, edge detection too has a number of variables which we cannot directly control. The camera quality, the illumination of the room(which leads to speckles), external air condition(moisture,fog),signals generated electronically by the circuits capturing the image, all of which introduce random noise in the image. We can always

apply smoothing operators to remove noise. However optimal filters working over all images are difficult to build. So are techniques for image differentiation, which like many other problems in computer vision is ill-poised in the sense that the stability, uniqueness and stability of the solution cannot be guaranteed. Image derivatives are very sensitive around noise and can generate vastly different results in the presence or absence of noise. It is difficult to generalize a solution for edge detection which can be used for variety of images. Differentiation is an ill-conditioned operator and smoothing results in data loss. Not all false edges generate from noise. It depends on the linking operation to classify the edges as true edges and false edges. This usually depends on a given threshold. Due to fluctuations in the value of this threshold for variety of images, it is not easy to generalize a universal edge detection algorithm. While noise edges can be removed using methods like hysteresis, phantom edges arising due to staircase noise. They form continuous curves and extends an authentic curve.

Another issue with edge detection is the scale at which it is applied. The scale of the image is often fixed by trial-and-error experiments and reused for all images. However an edge detector running at a particular scale does not detect all the edges. Hence we have to apply multiscale edge detection schemes that allows us to choose an edges detector after analyzing the image. However such systems can be computational intensive and not suited for small machines and embedded systems. The standards for edge detection are also not clearly defined. Thus deciding the optimality of an edge detector is difficult.

The figure shows running the Canny edge detection on two different images. While the first image pertains to the image of a koala the second is a grid of numbers. The first one might be used in something like an animal detection system, where zoologist can query the animal species from its pictures while the second one might be an input to an algorithm that arranges them in order like the 8 tiles problem. The first step in both the algorithm would be detect features. In the first case we might require the outline of the face of the koala while for the second case we might identify the blocks and the numbers contained in them. It can be seen from the figure that the conditions in which the images are taken is totally different. While the first one is taken in a real environment with different illumination and details of color in the koala, the second one is more computer generated. The states the importance of external factors on performance of the image. While our eyes our not able to distinguish the minute edges present in the koala image. However the Canny edge detection algorithm records the minute differences in the image itself and the result is obtained.

It might not be all that bad for edge detection to be so sensitive. For

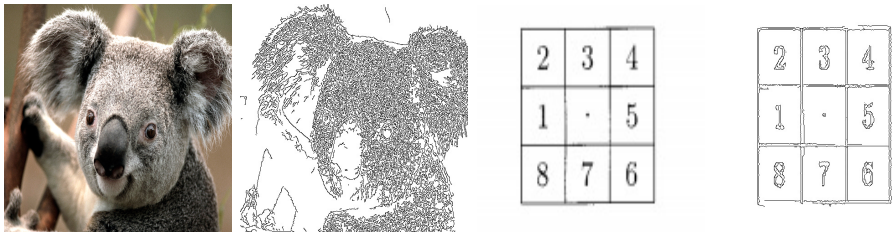


Figure 1.7: The images show the result of edge detection on two different images. While the first one pertains to a particular entity, perhaps captured by a person, the second one serves more as an input to a problem.

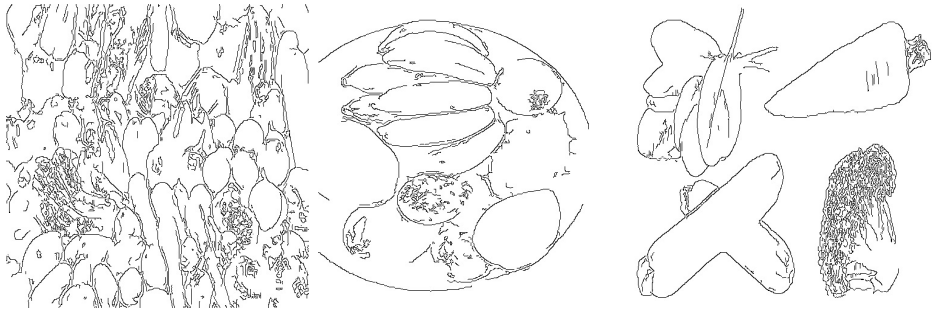


Figure 1.8: The images show the result of edge detection on three different types of images pertaining to fruits and vegetables. The different images contain different number of items kept together. The first one depicts the effect of edge detection on an image with a large number of objects. The second one represents the effect of edge detection on an image with few number of objects while the third one represents edge detection on an image with not many objects but they are clustered closely together

example, if an edge detection algorithm is run on the an MRI scan, it may help detect small variations in the region of brains resulting in better prognosis of the disease. However there is a downside too. If the edges detected are fake this might lead to a wrong prognosis which can be very fatal. This might be a reason why the current detectors are not applicable to automate such systems. Being sensitive to small images is not the only problem suffered by edge detectors. When there multiple objects in an object closely aligned with each other the detection may not produce good results. Consider the figure below. The figure shows the output of edge detection on three different images. The images may be the input of an automatic cataloguing system. As it can be seen the output varies as per the number of items in the image. The first images consists of a large number of objects in one image. The boundaries become less clear and hence not all objects can be identified. The second image has fewer objects, the results are better but still there is loss in vital edges that help identify the objects(bananas in

this case). The last image consists of fewer objects arranged seperately and hence the boundaries are detected nicely.

These are only some of the few examples of different types of input given to a computer vision application. One edge detector cannot successfully detect edges for all cases, neither can it detect all edges. Our eyes have evolved over millions of years to reach the perfection we have. And it too is dependent on illumination of an object and does it will still take some time to perfect this system. However we can still refine the outputs of the image. For example we can link gaps in the image using morphology and threshold unwanted small edges and maintain edges oriented along a strong direction.

Chapter 2

Literature Survey

Edge Detection is a very popular area of research in the field of image processing. It is fundamental step in several image processing step and hence has been studied extensively. However despite years of research it still holds a lot of potential for improvement and active research. A variety of edge detection algorithms have been devised which differ in their purpose and algorithmic properties[2][3][6]. Most of the methods in edge detection can be grouped into two categories, first gradient based methods and zero crossing or second gradient based methods. Canny[1] proposed an optimal edge detector based on the derivative of the Gaussian Filter, along with non-maximal suppression and double thresholding. Deriche[9] implemented a fast recursive implementation of Canny's algorithm. Maar and Hildreth[4] designed edge algorithms based on the second order derivative of the image.

More modern techniques include using mathematical morphology, wavelet transforms and fuzzy logic. Lei Zhai et al[8] presents a overview of all these methods. Wavelet transform is a powerful tool for time-frequency analysis, and allows analysis of local properties of functions in terms of their spectral properties at a particular time. Wavelets act similar to kernel used in Canny edge detection but are more selective in nature and hence provide better detection of edges. However it is not always easy to determine the wavelet function to be used. It is also used for detecting edges in medical images[16].Mathematical morphology provides an innovative way to detect edges. Structured elements of different shapes can be used to perform operations that dilate, erode, close or open an image. Deng Caixia et al[13]. proposed an improved edge detection algorithm using morphology, wherein they use two structural elements,one of size 3x3 and the other of size 5x5 and orient them along different directions. This gives them two sets of edges and then fuse the edges to generate the edge image. Senthikumarna N and Kirubakarn C[11] give the application of morphology in segmentation for MRI brain images. Research has also been carried out to determine the use

of concepts like fractal analysis and fuzzy sets for edge detection in particular applications.

There has been a lot of research on the Canny Edge detector itself. T Rupalatha et al[15] and Qian Xu et al propose the implementation of a distributed Canny Edge detector on Field Programmable Gate Array(FPGA) based architecture. Ogawa K has proposed an efficient Canny Edge Detection algorithm on CUDA(Compute Unified Device Architecture). Philip Worthington proposes an enhanced Canny Edge detection using a concept called curvature consistency. The algorithm tries to form regions of genuine curvature without over smoothing the genuine edges. This is done by adjusting the gradient estimates to reduce the difference between shape-index labels, proposed by Koenderink and van Doorn, of neighboring pixels. Ping Zhou et al[5] propose an improved canny edge detection algorithm, using Otsu method for adaptive thresholding. Other methods include Ant Based Approach proposed by Aleksandar Jevtic and Bo Li[7] for adaptive edge detection. The algorithm randomly distributed "ants" on the pixels of the image. Each ant will then compute the probability of moving in a certain direction based on the pheromone value of the neighboring pixels. There is also a negative feedback through pheromone evaporation. These steps are iterated in a loop till the maximum number of iterations is reached and eventually the ants settle along the edges of the image. Piotr Dollar and C.Lawrence Zitnick[10] propose edge detection algorithm using structured forests. This includes structured learning which classifies a sample by recursively branching left or right down the tree until a leaf node is reached. Mitra Basu[17] discusses the use of Gaussian Filter in edge detection and reviews several linear and nonlinear Gaussian Based Edge Detection methods.

The algorithms mentioned mostly consider detecting edges in gray scale images. There have been several approaches to detect edges in color based images. Ramakant Nevatia[28] was one of the first to do so and discusses color edge detection and its application in scene segmentation. Soumya Dutt and Bidyut Chaudari[18] discuss a color edge detection algorithm in the RGB color space. Their method processes the image and automatically selects a threshold to generate the edge map. They first transform the color image to a single channel pixel containing weighted values of the three RGB values and then apply a four directional color mask to compute color differences in four directions.The results were positive showing lesser false edges than Sobel or Laplacian operator. Ajay Mittal et al[25] have surveyed several methods to perform color edge detection. They classify methods as either synthetic or vector based methods. Shu-Yu Zhu et al[20] provides an analysis of edge detection in color images. They provide a comprehensive analysis of methods used in color edge detection. P.E Trahanias and A.N

Venetsanopoulos[27] discuss color edge detection using vector order statistics and shows how their minimum vector dispersion method responds better to edges with noise. Gwanggil Jeon[22] compares edge detection in different color spaces like the RGB, YIG and HSV color space.

There has been research for performing edge detection using machine learning. Palvi Rani and Poonam Tanwar[23] discuss a method for edge detection using normal networks and fuzzy logic. Some research is also focused on finding dynamic thresholds for edge detection. Edge detection based on genetic algorithm is discussed by Suchendra Bhandarkar[24]. They edge configurations are viewed as two-dimensional chromosomes with fitness values inversely proportional to their costs. Genyun Sun et al[29] discuss a method to seperating edges based on detecting boundaries between clusters. Rishi Rakesh et al[26] discuss statistical approach for thresholding in edge detection. They first estimate the image surface using bivariate smoother and then calculating the gradient vector at each pixel fitted on the surface. They use non maximal suppression and thresholding with hysteresis. Suryakant and Renu Dhir[19] propose an edge detection method using adaptive neuro-fuzzy inference system that trains data by analyzing the region around a pixel and using built in rules determine if the pixel is an edge point or not. Some algorithms have been designed to detect edges of specific objects only. Li-qin Jia et al[21] propose a fast randomized algorithm circle detection algorithm. The algorithm uses Canny edge detection and then randomly selects points from the circle and checks if their gradient passes through the center. An entropy based approach was suggested by Mohammed A.El-Sayed[30] discusses a algorithm for edge detection based on entropic threshold. The thresholding method computers the probability distribution of gray values in the image. For all the threshold values they compute the Tsallis entropy and select the threshold which maximizes the information measured between the two groups of pixels. They then apply the gradient mask on the image.

Many implementations of edge detectors discussed above apply Canny edge detection and use it as a base model to further refine its results. Canny has developed a method that has become a standard edge detector since its inception. While the traditional Sobel operator does not give good results for all images, the more advanced ones like neural networks and machine learning based take time to process and require a lot of training data. It can be seen that the Canny edge detector is a very popular and fast edge detector, having many practical applications and an algorithm that can be used in conjunction with other methods to provide better performance for edge detection in an image. Hence we explore methods to preprocess or post process data to generate better results.

Chapter 3

Methods Explored

Canny is one of the most popular edge detectors. It is innovative in the sense that it uses techniques likes non maximal suppression and hysteresis thresholding to improve edge detection. However there are certain factors that it can control. These include external factors like noise and illumination which greatly affects the performance of the edge detector. Many times it leads to detection of false edges. For example, consider the following image and the resultant image after applying Canny edge detection(threshold=75).

Figure 3.1: (a)Original Image

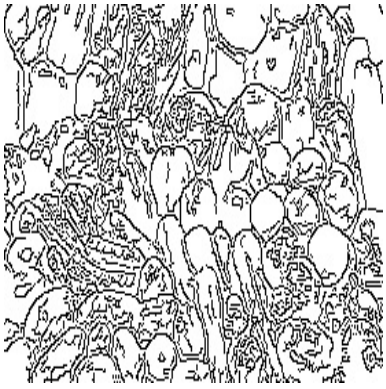
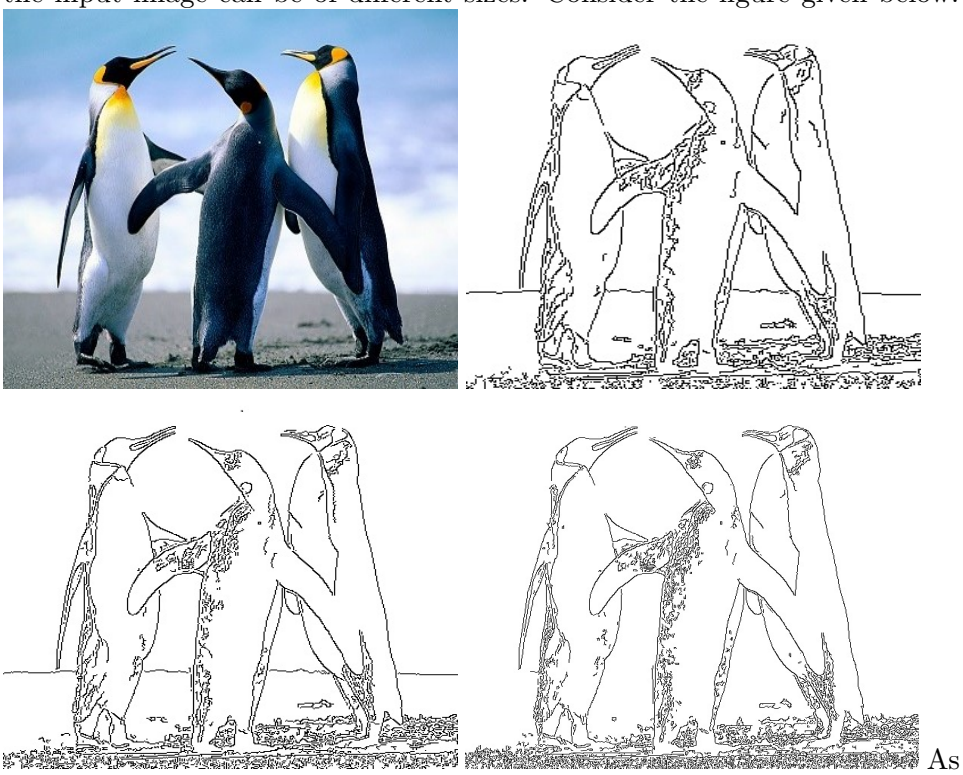


Figure 3.2: (b)Canny Edge Image

It can be seen here that for multiple objects closely spaced with one another, it is very difficult for the edge detector to perform optimally and leads to many false edges. Illumination problems create reflections in objects which are mistaken for edges and hence generate false edges. We hence try to explore methods that will process the image in certain way that will make the output generated by the Canny algorithm better.

3.1 Small Scale Edge Detection using Recursion

Every application requires the input to be resized to a certain scale, since the input image can be of different sizes. Consider the figure given below:



As

Figure 3.3: Size of image and output of the algorithm(Left to right from top)(a)Original Image of size 300x400 (b)Canny Edge Output of image resized to 300x300 (c)Canny Edge Output of image resized to 400x400 (d)Edge Output of image resized to 600x600

it can be seen from the images, Canny seems to work better with smaller size images. This might suggest that we may use Canny on a smaller part of an image and the join the results to provide better edge detection. Hence we can apply recursion to divide the image into subimages and apply Canny edge detection on that. Once done we can merge the results into a single image.

To see why the algorithm could work, we first observe that the algorithm requires less number of convolutions. Consider an image of size NxN and the Gaussian Filter of kernel size W. The number of convolutions required will be $(N - \frac{W}{2})(N - \frac{W}{2})$. Now consider the number of convolutions required when the image is divided into blocks. If the block size is S, each block would require $(S - \frac{W}{2})(S - \frac{W}{2})$ convolutions. Now there are $(\frac{N}{S})(\frac{N}{S})$ such

blocks and hence the total number of convolutions is given by $(S-\frac{W}{2})^2(\frac{N}{S})^2$. This can be reduced to $(N-\frac{WN}{2S})^2$. Since $\frac{N}{S} > 1$, we see that the number of operations have decreased.

The first step is convolution of the image with the Gaussian filter. Consider the Gaussian function G(x,y) and image signal I(x,y). Then the convolution can be given by:

$$H(x,y) = G(x,y) * I(x,y) = \int_{-W}^{W}\int_{-W}^{W} G(\tau_x,\tau_y)I(x-\tau_x, y-\tau_y)d\tau_x d\tau_y \quad (3.1)$$

where W specifies the band limit of the Gaussian filter used. Now for discrete images this can be written as

$$H(x,y) = \sum_{a=-W}^{a=W}\sum_{b=-W}^{b=W} G(a,b)I(x-a, y-b) \quad (3.2)$$

Then we take the derivative of the image using Sobel operator in both X and Y direction:

$$h_x(x,y) = \frac{\partial h}{\partial x}(x,y) = \sum_{a=-W}^{a=W}\sum_{b=-W}^{b=W} G(a,b)I_x(x-a, y-b) \quad (3.3)$$

where $I_x(x,y)$ denotes the partial derivatives wrt x. We can also find the derivative of Gaussian $G_x = xe^{\frac{-x^2+y^2}{2\sigma^2}}$. Since noise is randomly distributed throughout the image, applying Gaussian on small scale can help reduce more noise.

3.1.1 Basics of Recursion

Recursion is a method of solving a problem, wherein the solution to a problem depends on the solutions to the smaller instances of the same problem. In other word a problem can be divided into multiple independent problems(note that the problems are independent unlike dynamic programming). Recursion is part of the Divide and Conquer strategy. Recursion is usually of the following format:

$$f(x) = \begin{cases} g(x), & \text{if x satisfies the base case} \\ a * f(x/b) + c, & \text{otherwise.} \end{cases} \quad (3.4)$$

The time complexity of such algorithms is given by solving the time series equations:

$$T(n) = a * T(n/b) + O(n^k) \quad (3.5)$$

The solution to this problem is given by the Master's Theorem as follows:

$$T(n) = \begin{cases} O(n^{log_b a}), & \text{If a} > \text{b}^{\text{k}} \\ O(n^k.logn), & \text{If a} = \text{b}^{\text{k}} \\ O(n^k), & \text{if a} < \text{b}^{\text{k}} \end{cases} \tag{3.6}$$

3.1.2 Using Recursion with Canny Algorithm

As explained earlier, recursion can be used to perform edge detection on subimages and then join the results to obtain the entire image. The figure below how recursion will go about in the image:

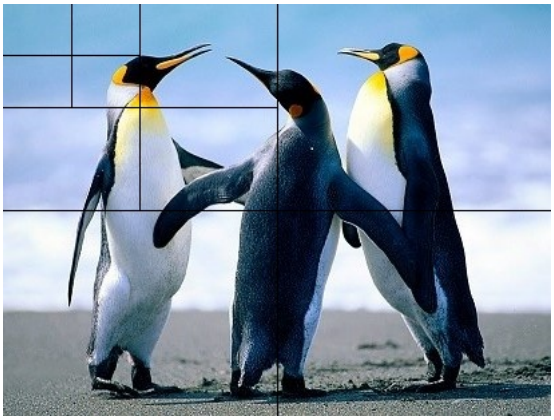


Figure 3.4: Using recursion an image will be divided into four quadrants and each quadrant will go about the same process

The algorithm can be generalized as follows:

EDGEDETECTION(Image):

1. Divide the image into four regions into S_i, i $\epsilon\{0, 1, 2, 3\}$
2. For each region do:
   (a) IF sizeof S_i is less than Base Case:
      i. Filter S_i and Perform Canny Edge Detection on it
   (b) Else Call EDGEDETECTION(S_i)
3. Join the results obtained from each region S_i
4. Return the joint image

3.2 Edge Filtering by determining Conditional Probability of a point being an Edge

The edge detector provided by Canny gives good performance in most cases. However due to noise and intensity changes, the algorithm detects edges that are irrelevant, small and hinder the further image processing activities. However post processing the edge image can allow us to remove unnecessary edges. For every pixel detected by the Canny algorithm we analyze its surroundings. We then measure the amount of order in the neighborhood of the pixels. By order we mean if the pixels can be aligned along a particular line(which is expected from edges). We use regression to fit a line along the points given and then check how many actually fall on the line. This will determine the Probability whether the pixel is part of an edge or not. However it cannot be assumed that since Canny edge detector classifies the pixel as an edge point that it is an edge. There is a probability that the edge detected by canny algorithm is not an actual edge.Let $P_{(i,j)}$ denote the Probability of point (i,j) being an edge point. Let P_{TT} denote the probability that Canny algorithm classifies (i,j) as an edge point under the condition that (i,j) actually is an edge point. Similarly let P_{FT} denote the probability that the Canny algorithm does not classify (i,j) as an edge point under the condition that (i,j) is actually an edge point. Then the probability that (i,j) is an edge point under the condition that Canny classifies it as an edge point Pr is given using Bayes theorem as:

$$Pr = \frac{P_{TT}P_{(i,j)}}{P_{TT}P_{(i,j)} + P_{TF}(1 - P_{(i,j)})} \tag{3.7}$$

Note that P_{TF} ie Probability that Canny algorithm classifies a point (i,j) as an edge point under the condition that (i,j) is a false edge is simply $1 - P_{TT}$. Similarly for every black pixel we can find the probability if it is a true edge under the condition that canny detects it as a false edge. However this is time consuming and significantly slows down the algorithm. Hence we only analyze the white pixels available. Note that the probability $P_{(i,j)}$ is obtained by analyzing the neighborhood of the pixel. We try to fit a line through the points. We then check how many points actually lie on the line and divide it by the total number of points. Note that we analyze only small neighborhoods where the edge points lie more or less on a straight line.

The process will filter out edge points. However it will create discontinuities. We must then perform edge linking. Edge linking is the process of joining set of discontinuous edges. The process is simple. We collect the endpoints of each edge and the continue along the direction the edge was oriented in until we find another edge point. We also remove edges with small edge lengths so that they do not give rise to unnecessary edges. The figure below shows some blocks and their probabilities. The size of block used is 6x6.

Figure 3.5: (a)Probability = 0.48

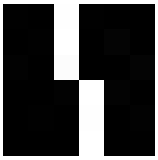


Figure 3.6: (b)Probability = 0.53

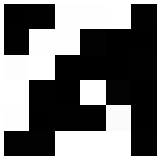


Figure 3.7: (c)Probability = 0.38

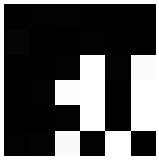


Figure 3.8: (d) Probability = 0.40

Figure 3.9: Probability of different blocks

3.2.1 Polynomial Regression

Polynomial Regression or Curve fitting involves capturing the trend in data by assigning a single function across the entire range of data. In simple linear regression, we try to fit the data along a line $y = a_0 + a_1 x$. For small neighborhoods we can fit the points along a line. However for larger areas, we must consider higher degree polynomials or curves to fit the data to. In general we can model the expected value of y as an n^{th} degree polynomial, which can be written as $y = a_0 + a_1 x + ... + a_n x^n$. From the polynomial regression model we can see that $y_i = a_0 + a_1 x_i^1 + ... + a_n x_i^n$ for i ϵ 1,2...n. We can write this in matrix form as:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & ... & x_1^m \\ 1 & x_2 & x_2^2 & ... & x_2^m \\ 1 & x_3 & x_3^2 & ... & x_3^m \\ . & . & . & ... & . \\ . & . & . & ... & . \\ 1 & x_n & x_n^2 & ... & x_n^m \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ . \\ . \\ a_m \end{bmatrix} \tag{3.8}$$

Hence we can write it as $Y = XA$ and hence we can solve for A is $(X^T X)^{-1} X^T Y$.

3.2.2 Edge Linking

Edge detection algorithms might return broken boundaries due to factors like noise and non-uniform illumination. Thus we have to follow edge detection with edge linking. We have to assemble the edge pixels into meaningful boundaries. Edge linking can be done locally or globally. In local processing we analyze neighborhood of the pixel and we establish similarity between points. We compute the similarity between two pixels in a neighborhood from the Sobel operator. The Sobel operator returns both the horizontal

and vertical derivatives G_x and G_y. We then set two thresholds, gradient threshold T and angular threshold A. For two pixels (x,y) and ($x^{'}$,$y^{'}$) in each others neighborhood, the algorithm will classify them as similar and link them. For global processing, we can use Hough transform. Hough transform, which uses a voting mechanism to estimate parameters(used in regression) to be selected, and in turn reduces the number of parameters required for regression. The voting is done by points on a curve. Consider a linear line

$$y = mx + c$$

. Here x and y represent observed values while m and c represent the parameters. The line can also be written as $c = -xm + y$. Here c and m represent the variable and x and y the parameters. All the (x,y) on the line in the x-y space represents a family of lines in m-c space. Thus if we are interested in the straight line that best fits given n-points, we can work in the m-c domain instead of the x-y domain and apply hough tranform. The point through which maximum number of lines pass through gives the m and c of the best fitting line. Hough transform becomes inefficient as the number of parameters increase, especially for 2-D curves. The performance can be improved by using gradient information on the boundary of the image to reduce the work in the parameter space. Consider the curve to be fitted is a circle given by the following two equations:

$$(x - a)^2 + (y - b)^2 = r$$

. Then converting it into parametric form

$$x = r\cos\theta + a$$

$$y = r\sin\theta + b$$

Thus a and b can be written as: $a = x - r\cos\theta$ and $b = y - r\sin\theta$. Thus we can record the gradient angles θ_i at edge point(which is precomputed during edge detection) and then measure the distance r from a fixed base point using the euclidean distance. This can be generalized for all types of curves.

We here use a localized algorithm. For each edge point, we find the approximate direction of the edge in one of the eight directions $(0^o, 45^o, 90^o, 135^o, 180^o, 225^o, 270^o, 315^o)$. We then go on finding edges till we meet another edge point. The algorithm is as follows:

1. Perform Edge Detection using Canny algorithm on an image.
2. Find the endpoints of Edges detected and store them in a list L.
3. While L is not empty:

(a) Let L_0 indicate the first element of the List which stores the coordinates (i,j). Analyze the 8-neighborhood of L_0 and find the direction (from the 8 directions) of the edge. This can be done by finding an (i,j) which is white and moving along (i_r,j_r), where (i_r,j_r) denotes the reflection of (i,j) from the central pixel.

(b) If (i_r,j_r) is not in the list L:

i. Add (i_r,j_r) to the list L. Mark pixel (i_r,j_r) as white.

(c) Remove L_0 from list L

4. End.

3.2.3 Algorithm for the Edge Filtering Technique

The algorithm for the process described above is given below:

1. Perform Edge Detection using Canny algorithm on an image.
2. For each point detected as an edge by Canny algorithm:
   (a) Analyze the neighborhood around the point. Using regression, determine a line ax+b which best fits all the points in the neighborhood which are white. Once we determine a and b, we find the number of points (x,y) for which y = ax+b. We then compute Probability $P_{(i,j)}$.
   (b) We then determine the probability Pr mentioned in equation 3.4. If Pr $>$ 0.5 the we retain (i,j) else we set pixel (i,j) as black.
3. Remove small edges and perform edge linking.

3.3 Results

The initial results of the two algorithms have not been quite upto the mark. While the recursive method works well in removing noise from blocks. However the contour based method does not work very well and the quality of edges is degraded from that of canny edge detection algorithm. Nevertheless it does remove noisy edges. The initial results are shown. The results show that the recursive method employed significantly reduces the internal noise and small edges present in the canny edge detector. However the algorithm might lose out some edges, though it maintains the overall outline of the significant objects. Edge linking must be performed on the result obtained and this will enhance its performance. The Edge Filtering method removes the small edges while maintaining the edges detected by Canny. However the linking process generates edges which is not significant.

The images show that the recursive algorithm effectively filters out small edges and gives an outline of the objects in the image. However there are frequent loss of edges. Operations like dilation can fill the gaps or perform edge linking but this may introduce unnecessary edges as seen in the last figure. The Edge Filtering algorithm preserves all the major edges, but does not filter the small edges effectively.

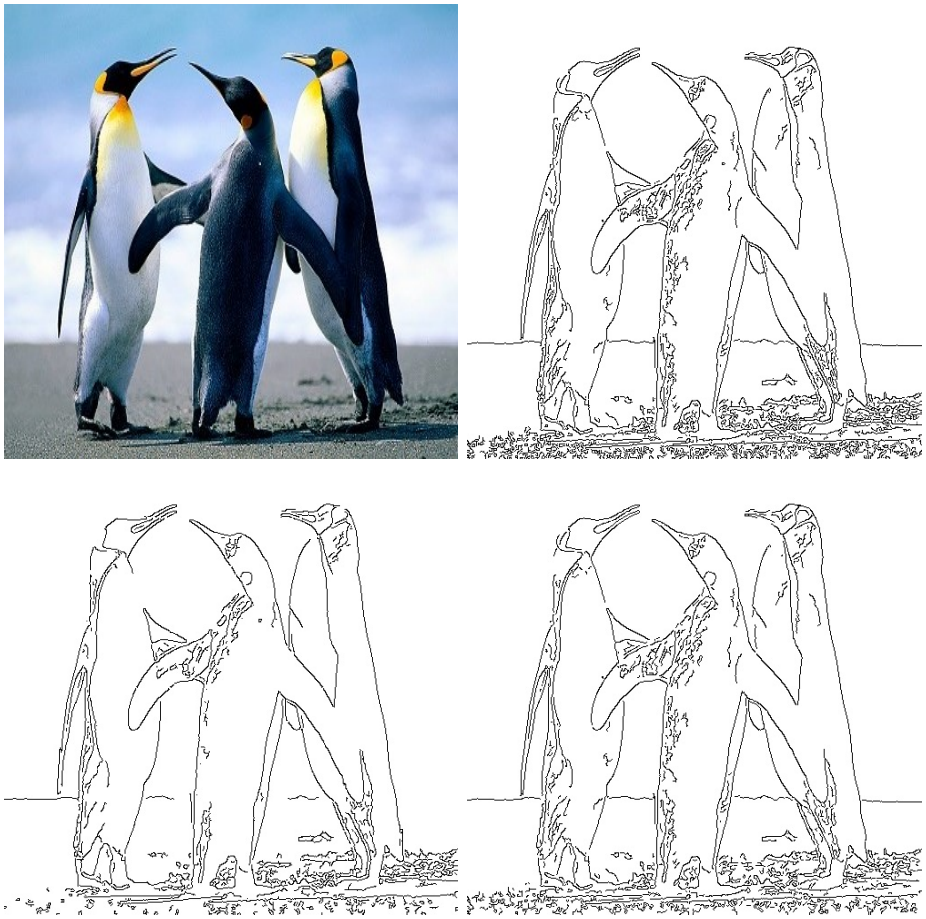


Figure 3.10: From Left to Right: (a)Original Image (b)Result from Canny Algorithm (c)Result by implementing Recursion and Canny algorithm (d)Result by using Edge Filtering method. As it can be seen, the Canny algorithm detects a lot of small edges arising due to reflection on the back of the penguins. The recursive method removes them significantly but also removes the boundaries of the penguins. The Edge Filtering method reduces maintains the boundaries but does not reduce the small edges significantly.

Figure 3.11: From Left to Right: (a)Original Image (b)Result from Canny Algorithm (c)Result by implementing Recursion and Canny algorithm (d)Result by using Edge Filtering method. The recursive method provides a good outline of the image while the edge filtering method also gives an improved result from the original Canny image.

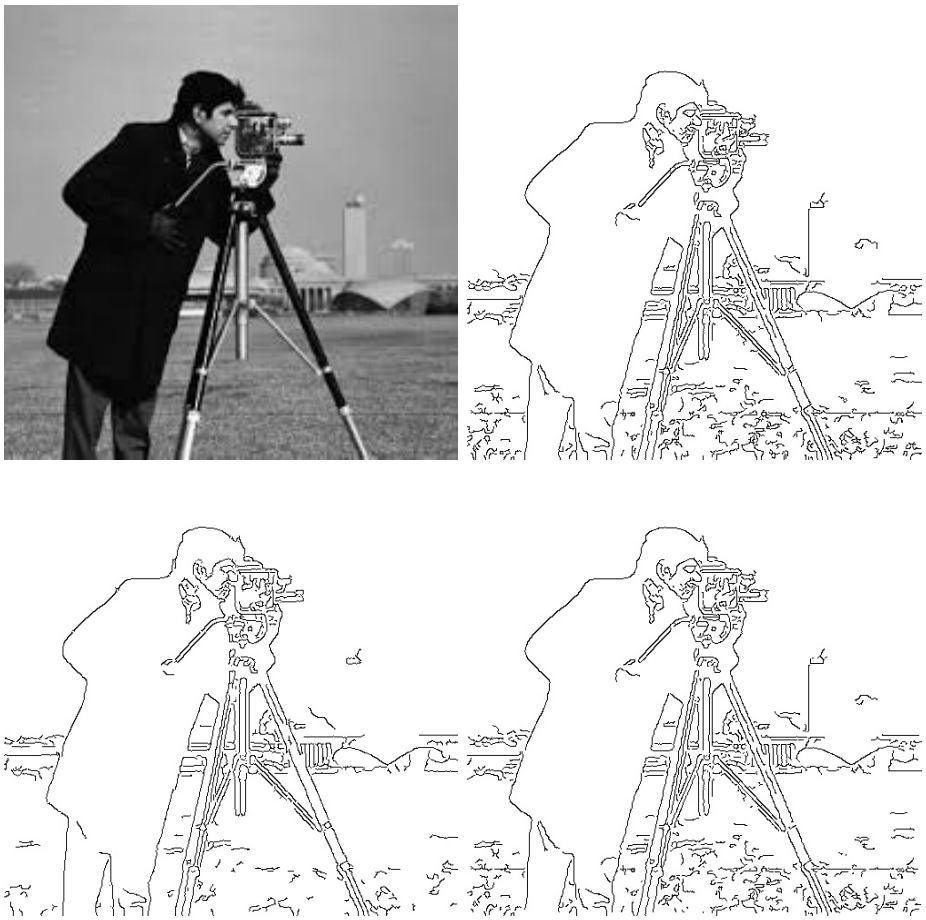


Figure 3.12: From Left to Right: (a)Original Image (b)Result from Canny Algorithm (c)Result by implementing Recursion and Canny algorithm (d)Result by using Edge Filtering method. The performance of Canny and the Edge Filtering method is almost the same.

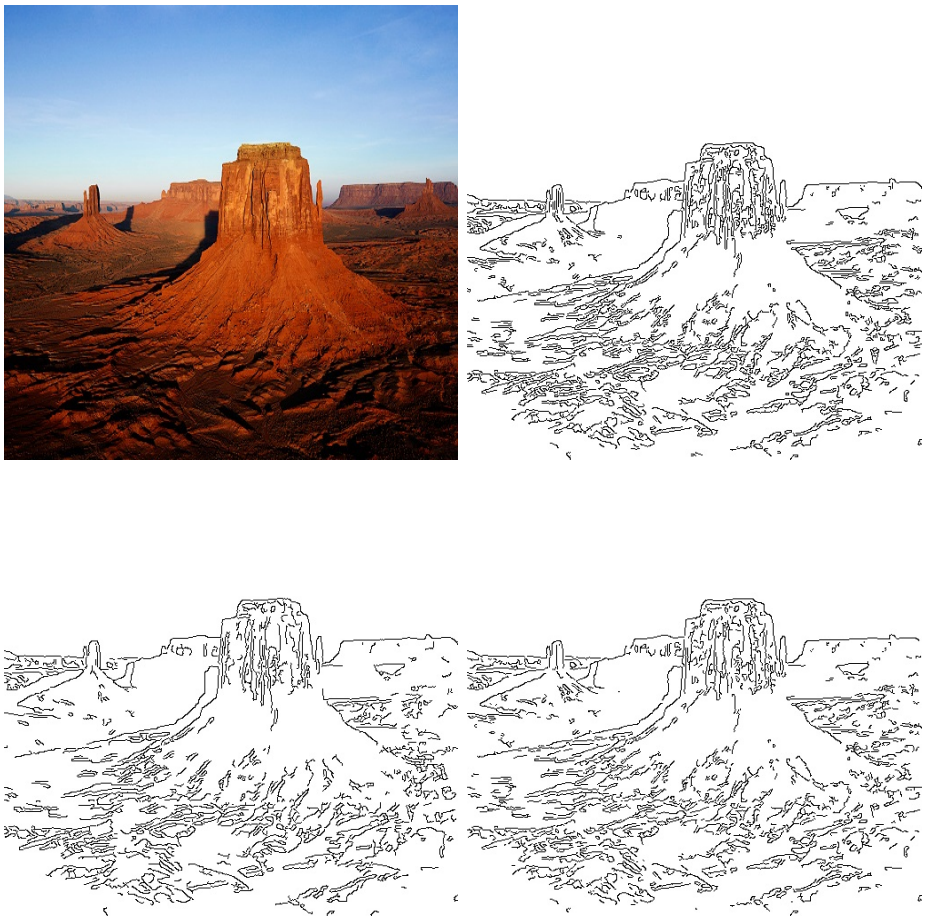


Figure 3.13: From Left to Right: (a)Original Image (b)Result from Canny Algorithm (c)Result by implementing Recursion and Canny algorithm (d)Result by using Edge Filtering method. As seen the Recursive method shows significant improvement over the other two.

Figure 3.14: From Left to Right: (a)Original Image (b)Result from Canny Algorithm (c)Result by implementing Recursion and Canny algorithm (d)Result by using Edge Filtering method. Here the recursive method removes the outlines of several objects and may not be helpful for multiple objects.

Chapter 4

Analysis of the Results

The results in the previous chapters show that the Canny Edge Algorithm is still the most optimal algorithm. The performance of the two methods tried do not bring significant changes to the performance of Canny. However some small changes are see and should be analyzed.

Let us first consider the picture of the penguins given and see the notable differences between recursively applying Canny and the usual Canny method. Consider Fig 4.1. Here we have marked four regions 1,2,3 and 4. Each region distinguishes the two images in some manner. There are several other regions that differ, however we focus on these three for now. It can be seen that the original Canny maintains the edges defining the shape of the penguin more or less intact. As seen from region 1 and region 4, lines are maintained by the original Canny. This is a result of the hysteresis linking used by the Canny algorithm which links the edges. However if we see regions one and four in the output after recursively applying Canny for small scale, we see that the line remains broken there. Other missing lines include the head of the leftmost penguin. This is interesting result which helps us visualize Canny without the hysteresis linking. It is interesting to now analyze regions with a lot of clutter or unwanted edges like region 2 and region 3. The absence of such clutter is due to applying the Canny edge detection algorithm at a small scale. Applying Gaussian on a small scale allows small scale filtering as mentioned before. Hence those edges are treated as noise and removed. The other thing that can be noticed is that the noisy edges were mainly results of bad hysteresis edge linking. Since the recursive implementation of the algorithm does not suffer from any such clutter it must be because of improper edge linking. Hence the Canny algorithm detects most of the true edges in the first phase itself. It can also be observed that linking at a small scale results in fewer false edges than large scale linking. However there is a tradeoff as the small scale linking seems to miss out on certain edges. Hence it might be useful to apply small scale edge detection

first and then linking the edges together.

Though it should be noticed that recursively applying the edge detection algorithm is not much different from applying the edge detection to the entire image. The still convolves the entire image and hence reduces noise uniformly. It should also be observed that the more times the Gaussian blur is applied the better the outline generated by the Canny algorithm is, however oversmoothening can lead to loss of vital edge information and hence true edge of the image. It is interesting to see the effects of the size of the block(recursion base case) and kernel size on the image. If we apply differ-

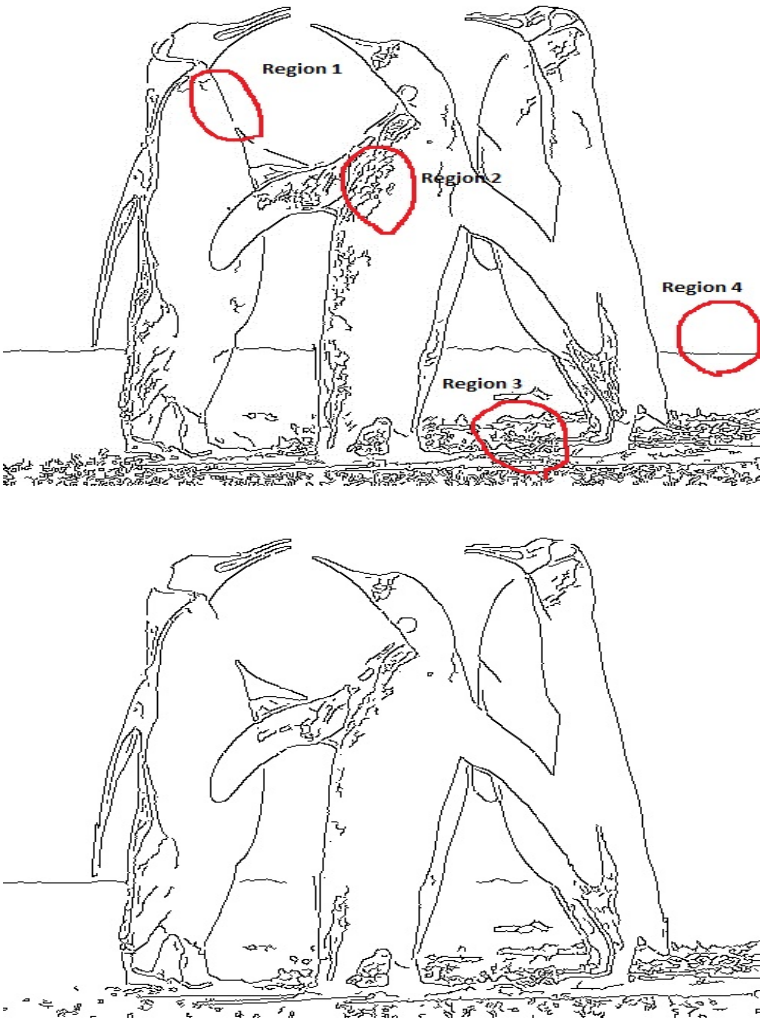


Figure 4.1: From Left to Right: (a)Canny output of the Penguin Image (b)Recursive Canny Output of the Penguin Image.

Figure 4.2: Result for using different kernel sizes on the base case block.(a)Kernel size=5 (b)Kernel size=7 (c)Kernel size=9 (d)Kernel size =11. Size of block is 64x64. Size of image is 512x512

ent size kernel we get the following results as shown in Figure 4.2. It can be seen that as the kernel size increases, the outline(the important edges) become more significant and the false edges decrease. However once it goes

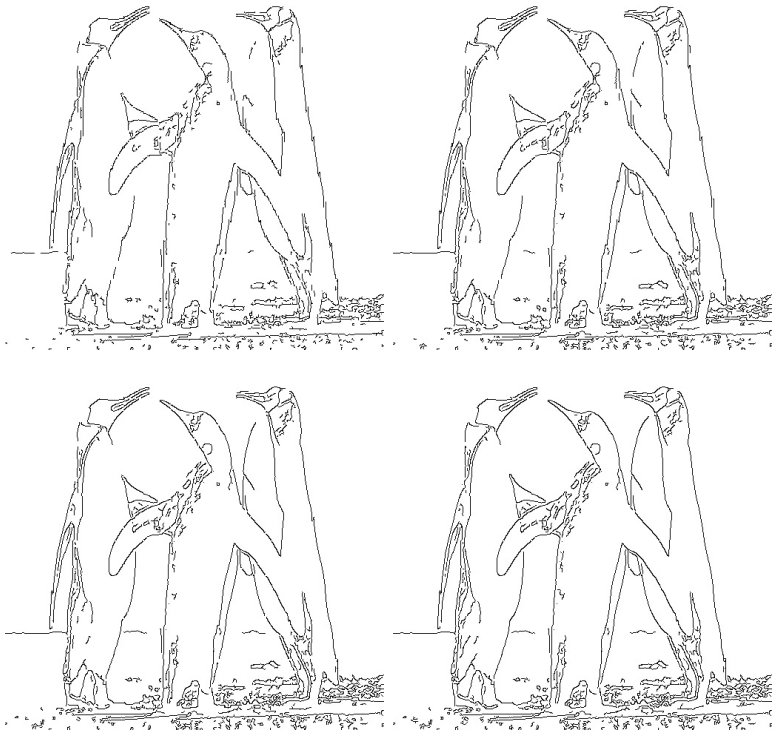


Figure 4.3: Result for using different block sizes used in the the base case of recursion.(a)Block size=8 (b)Block size=16 (c)Block size=32 (d)Block size =64. Size of Gaussian kernel is 3x3. Size of image is 512x512

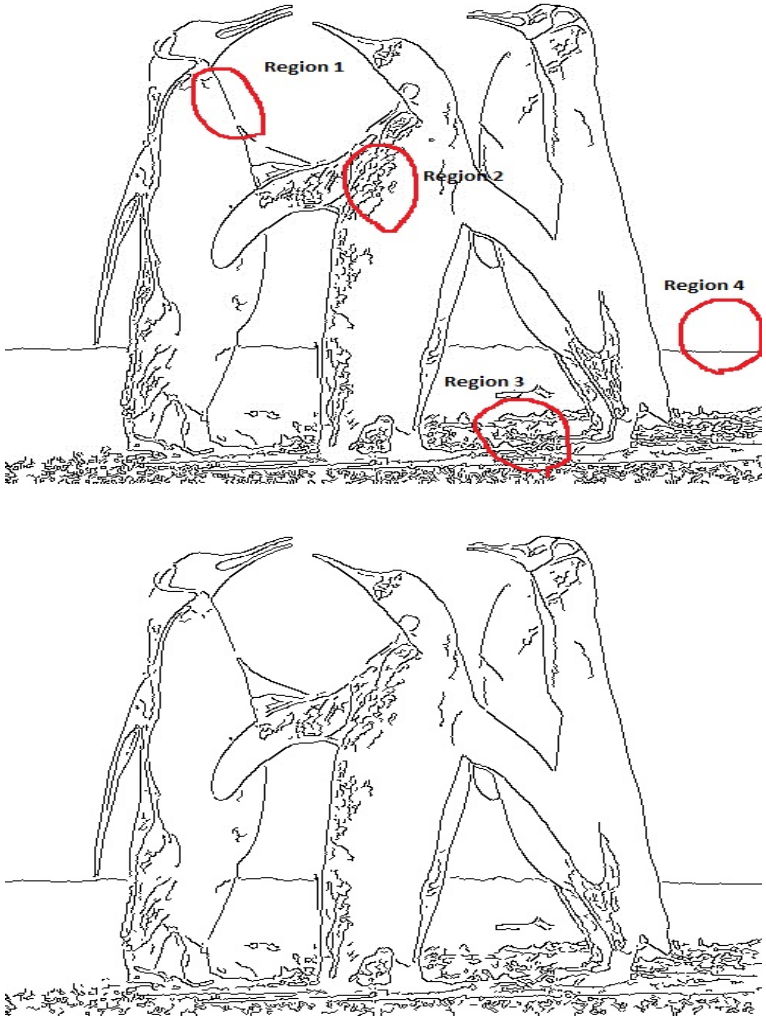


Figure 4.4: (a) Output of Original Canny Image (b) Output of Canny Image after applying the refinement algorithm.

beyond size 7, there is a significant loss in the actual edges. Also the computation time for larger kernels keeps on increasing(i.e to find an appropriate Gaussian filter for a particular size). In figure 4.3 we analyse the effect of size(image block size used in the base case) on the output produced by the algorithm. For smaller block size, the output is jittery and not smooth or continuous as seen for size = 8. For larger values of block size however the removal of noisy and false edges reduces, leading to a similar performance to that of the original Canny algorithm. Hence block size of 32 and kernel size of 5 improves the quality of output generated. For larger block size the performance suffers since the results are not filtered at the appropriate small

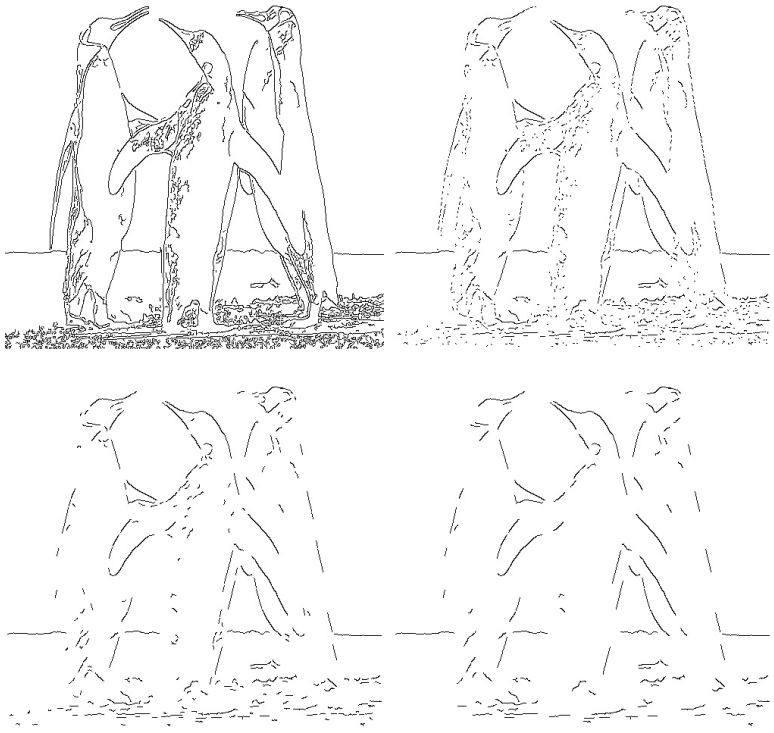


Figure 4.5: Outputs of the algorithm after refining the Canny image. We set the length threshold (b)Original Canny image (b)Length threshold =1 (c)Length threshold =5 (d)Length threshold =10

level, while for smaller block size the recursion leads to a jittery output with the edges appearing mismatched. Larger kernel size definitely smoothen the image, but it affects the time taken for convolving the image with the kernel and hence may not be useful for practical purposes.

Let us now analyze the performance of the second algorithm wrt to the orginal one. For the same penguin image, the outputs are shown in Figure 4.4. From the figure, it can be seen that the two images do not differ significantly. The chief cause of this is because the algorithm later uses the original Canny image to link edges. The algorithm unlike the previous algorithm maintains the true significant edges in the Output. Region 1 and Region 4 which was not maintained in the output of the recursive method we discussed. The algorithm shows strong retention for edges which are straight. This is because the algorithm determines the probability of an edge based on the number of points that fit on the line(with a small error correction). Hence lines as in Region 1 and Region 4 which are free from

clutter are maintained as they are. Region 2 and Region 4 however do not show much improvement from the original image. This is not a fault of the refinement process. Rather the refinement process successfully removes them. We see this in Figure 4.5. The figure shows the output of the refinement step for different threshold lengths i.e the minimum length of edge accepted by the algorithm. As the threshold increases we see that the more significant edges(describing the outline of the image) are retained. For example the line in Region 4 is retained. However when we try to link the edges. We generate a result similar to the original Canny. This is a fault of the linking algorithm. We use the original Canny algorithm for linking the broken edges. Hence the output generated is not much different. It should be noted however that Region 2 and 3 of the refined output are less dense, nonetheless the outputs are not good enough.

The algorithm has a lot of interesting parameters we can analyse like the minimum acceptable probability(probability threshold). We observe the outputs in Figure 4.6. It can be seen that probability threshold simply

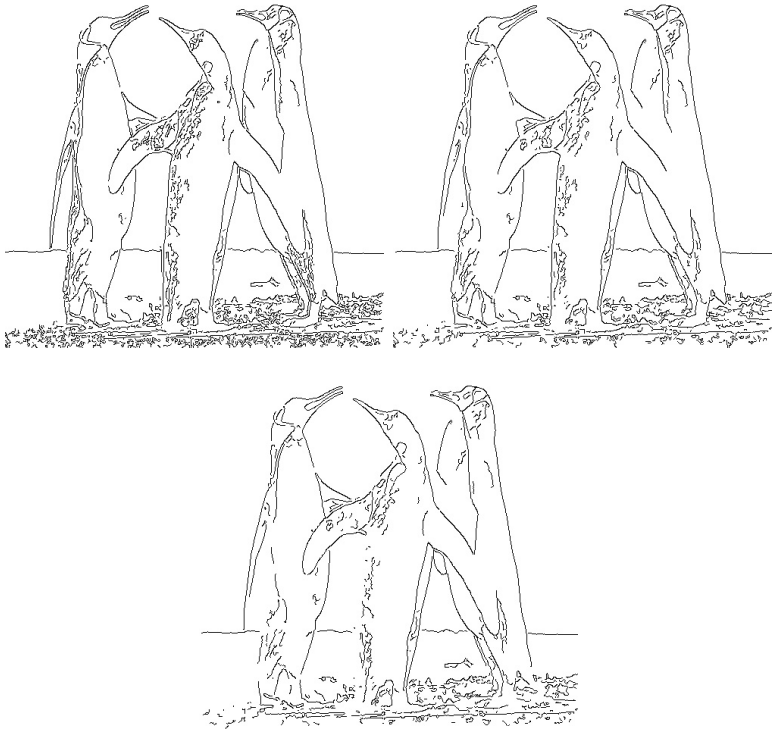


Figure 4.6: Outputs of the algorithm after refining the Canny image wrt different probability thresholds (a)Probability threshold =0.4 (b)Probability threshold =0.6 (c)Probability threshold =0.7

acts as a filter for removing edges. If we relax the constraint we get more edges, while as we increase our threshold the number of edges become less. It should be seen that strong step edges (lines)are retained in any case, showing the algorithm returns high probability for nearly straight lines. In case we allow thresholds very small the output will be similar to that of the Canny image. If we keep the threshold high we get an output consisting of very few edges. Hence we keep it at 0.5. Another important parameter to be considered is the truth value of the Canny Edge Detector. The algorithm determines the likelihood that a point is an edge using conditional probability based on whether the original Canny algorithm detects an edge. We see the effect of variety of probability values in Figure 4.7. As seen the probability of the Canny determines the how many edges are detected. It should be noted that this does not preserve the straight lines. This is because though the probability of the point being an edge point is high, we assume the output of the Canny algorithm itself has very small chance of being true and hence reduces the overall probability. Choosing a suitable

Figure 4.7: Outputs of the algorithm after refining the Canny image wrt different probability conditional probabilities of the Canny algorithm detecting an edge if it is a true edge (a)Canny truth value=0.4 (b)Canny truth value =0.6 (c)Canny truth value=0.7

Figure 4.8: Outputs of the algorithm after refining the Canny image wrt different probability conditional probabilities of the Canny algorithm detecting an edge if it is a true edge (a)Window size = 4 (b)Window size = 14 (c)Window size=18

probability value is hence important. Choosing a large value of the Canny algorithm leads to detection of false edges which would have been ignored, while choosing small values of truth value leads to removal of edges. Hence considering the optimality of the Canny algorithm we set it to around 0.7.

The last parameter we focus on is the size of neighborhood considered to find the probability that a point is an edge. The outputs for variety of window sizes is shown in Figure 4.8. Large window sizes require more time to process. Another problem with choosing large window sizes is that the edges cannot be approximated as lines rather are curves which will take more processing time. Small window sizes however cannot effectively judge the neighborhood and hence will lead to no improvement. Hence we choose a window size of around 6.

From the above it can be seen that the performance of the algorithms largely depends on the parameters we choose. They must be analyzed carefully. It is unlikely that one particular configuration will yield the best results but the best performance can usually be achieved in a certain range of values.

Chapter 5

Future Work

The initial results show that the algorithms are still not as sophisticated as the Canny edge detector. However they show encouraging results in eliminating noisy edges. The algorithms itself must be analyzed in further details to explain them better. For example we have taken the probability of Canny marking a true edge as 0.7. There is no defined standard on the accuracy of Canny algorithm. The criterion for correctness must be defined for the algorithm and after careful analysis the probabilities can be defined. A variety of entropy models like the Tsalli's model should be further explored and refined to suit the application. Recursively applying Canny on small areas of images removes a lot of noise while keeping the boundaries of the significant objects intact. However the blocks are treated independently rather than connected(i.e they are not convoluted) and hence generate discontinuities in the resulting edge maps. The size of the block is chosen using trial and error. Effects of increasing or decreasing the block size, and the threshold parameters of the Canny edge detector will be studied. The effects of other filters like median filters will also be studied to find the filter that gives the best results for small scale. It is will also be our effort to precisely define how the algorithms work, using mathematical formulations. This will lead us to better the algorithm and improve its performance.

The color space of the image is an interesting area to research. Edge detection algorithms affect each color space differently. The HSV color space is useful to provide cues on illumination of an object and hence can be manipulated to remove the speckles detected by the edge detection algorithm. Interconversion from one color space to another might aid filtering noise and hence better edge detection itself. Effects of maintaining hue and saturation constant will be studied and quantified. Machine learning holds great prospects for improving edge detection. The size of the neighborhood used to train the data must be kept flexible to generalize the algorithm. Edge Linking algorithms used are very basic. The algorithm must be refined to al-

low only significant edges to be drawn rather than small noisy edges. There are several areas that can be explored to improve edge detection, such as thresholding parameter of the Canny edge detector. Histograms of the image show interesting properties about the image and can be used to apply adaptive thresholding for iteratively improve performance of the edge detectors. The neighborhood of the pixel can also be explored and devise an algorithm similar to the voting mechanism to distinguish noisy edge points from true edge points.

The process of finding probability of a point being an edge point can be further improved. Polynomial regression though useful is time consuming as the block size increases. Though polynomial regression is effective for finding straight lines, as the block size increases lines cannot successfully represent the distribution of points and higher order curves are necessary. The process of finding 2D entropy and probability must be studied further. Edge linking algorithms still depend on a reference picture to join edges. Research on edge linking will help better results of all edge detectors and reduce the number of false edges significantly. While we discussed refining results from standard edge detectors research in fields like morphology and fractal analysis can help better the process of edge detection itself.

While we have focused on images of a certain size itself, the edge detectors performance will be quantified for different sizes and a multiscale approach to designing edge detectors will be studied. Texture and color also play an important part in defining an image and might hold cues as to improvement in the edge detection schemes. There are a lot of variables associated with the algorithms like threshold value, standard deviation of the filters used etc. All these variables contribute to the algorithms performance in their own way and hence testing the algorithms for different values of the variables is crucial. The tests might reveal values of variables that improve the results of the algorithms while may also help reduce the computation time. There will be tradeoffs however and such effects must be taken into consideration. Pattern recognition algorithms and randomized approach like ant based algorithms will be studied in conjunction with these algorithms and their performance will be analyzed.

Chapter 6

Conclusion

Hence we have analyzed several ways to improve the performance of Canny Edge Detection by supplementing it with different functions. It can be seen why the Canny algorithm is considered so popular and optimal. While the improvement in performance is not significant, the results show that these algorithms on further refinement can help better the output of the system. Edge Detection is one of the most important steps in any computer vision process. Its performance directly influences the outputs of the next stages and hence it is important topic for research. The algorithms are still in rudimentary forms and can be improved. If the image is taken in controlled environment a number of factors can be taken into account. A better camera lens and adjusting the illumination of the image increase the chance of detecting true edges and decreasing the probability of marking false edges. Though the algorithms are simple, they seem to remove noise and false small edges generated due to speckles while retaining the significant edges. There is a lot of future work that can be done on the algorithms. It is necessary to quantify the performance of these methods and apply them on a large data sets of images.

The field of edge detection is extremely important. If we wish to achieve true automation in our existing systems, the first step is edge detection and its performance are of utmost importance. The next generation of technology are smart devices, like refrigerators having inventory management, item tracking at home, age detection security system, automated topology system to identify hills, trees and resources, and all depend on how well they first identify objects in the image. Edge detection is one of the most research fields in computer vision. While a general solution to the problem is difficult to obtain we hope that algorithms to process the edge maps detected will improve its performance.

Bibliography

[1] John Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis And Machine Intelligence*, Vol.PAMI-8, NO.6, November 1986

[2] G.T.Shrivakshan, Dr.C Chandrasekar," A Comparison for various Edge Detection Techniques used in Image Processing",*International Journal of Computer Science Issues*,Vol 9,Issue 5,No 1, Sept 2012

[3] N. Senthilkumaran, "Edge Detection Techniques for Image Segmentation-A survey of soft computing approach",*International Journal of Recent Trends in Engineering*, Vol. 1, No.2, May 2009

[4] D.Marr, E.Hildreth,"Theory of Edge Detection",*Proceedings of the Royal Society of London,Series B,Biological Sciences*,Vol.207,No.1167,pp.187-217,Feb 29,1980

[5] Ping Zhou, Wenjun Ye, Yaojie Xia , Qi Wang,"An Improved Canny Algorithm for Edge Detection",*Journal of Computational Information Systems 7*,2011

[6] Rashmi,Mukesh Kumar,Rohini Saxena,"Algorithm and Technique on various Edge Detection: A Survey",*Signal and Image Processing:An International Journal*,Vol.4,No.3,June 2013

[7] Aleksander Jevatic, Bo Li, "Ant Algorithms for Adaptive Edge Detection"

[8] Lei Zhai, Shouping Dong, Honglian Ma,"Recent Methods and Applications on Image Edge Detection",*2008 International Workshop on Education Technology and Training and 2008 International Workshop on Geoscience and Remote Sensing*

[9] Rachid Deriche,"Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector",*International Journal of Computer Vision*,pp.167-187, 1987

[10] Piotr Dollar,C.Lawrence Zitnick,"Structured Forest for Fast Edge Detection",Microsoft Research

[11] N.Senthilkumaran, Kirubakaran C,"A Case Study on Mathematical Morphology Segmentation form MRI Brain Image",*International Journal of Computer Science and Information Technologies*,Vol.5(4),2014

[12] Neeta Nain, Vijay Laxmi, Ankur Kumar Jain, Rakesh Agarwal,"Morphological Edge Detection and Corner Detection algorithm using Chain Coding",*IPCV'06*

[13] Deng Caixia,Chen Yu,Bi Hui,Han Yao,"The Improved Algorithm of Edge Detection Based on Mathematics Morphology",*International Journal of Signal Processing, Image Processing and Pattern Recognition*,Vol.7,No.5,pp.309-322,2014

[14] J.Mehena, "Medical Images Edge Detection Based on Mathematical Morphology"

[15] T.Rupalatha, C.Leelamoham, M.Sreelakshmi,"Implementation of Distributed Canny Edge Detector on FPGA",*International Jouranl of Innovative Research in Science , Engineering and Technology*,Vol.2,Issue 7,July 2013

[16] J.Petrova,E.Hostalkova,"Edge Detection in Medical Images using the Wavelet Transform"

[17] Mitra Basu,"Gaussian-Based Edge-Detection Methods-A Survey",*IEEE Transactions On Systems,MAN and Cybernetics-Part C:Applications and reviews*, Vol.32,No.3,August 2002

[18] Soumya Dutt,Bidyut B.Chaudhuri,"A Color Edge Detection Algorithm in RGB Color Space",*International Conference on Advances in Recent Technologies in Communication and Computing*, 2009

[19] Suryakant,Renu Dhir,"Novel Edge Detection Using Adaptive Neuro-Fuzzy Inference System",*International Journal of Advanced Research in Computer Science and Software Engineering*, Volume 2, Issue 5,May 2012

[20] Shu-Yu Zhu,Konstantinos N.Plataniotis, Anastasios N.Venetsanopoulos,"Comprehensive Analysis of Edge Detection in Color Image Processing",*Optical Engineering*,Vol.30,No.4, April 1999

[21] Li-qin Jia,Cheng-zang Peng,Hong-min Liu,Zhi-heng Wang,"A Fast Randomized Circle Detection Algorithm",*4th International Congress on Image and Signal Processing*, 2011

[22] Gwanggil Jeon,"Measuring and Comparison of Edge Detectors in Color Space",*International Conference of Control and Automation*,Vol.6,No.5,2013

[23] Palvi Rani,Poonam Tanwar,"A Nobel Hybrid Approach for Edge Detection",*International Journal of Computer Science and Engineering Survey(IJCSES)*,Vol.4,No.2,April 2013

[24] Suchendra M.Bhandarkar,Yiqing Zhang and Walter D.Potter,"An Edge Detection Technique using Genetic Algorithm-Based Optimization",*Pattern Recognition*,Vol.27,No.9,1994

[25] Ajay Mittal,Sanjeev Sofat,Edwin Hancock,"Detection of edges in Color Images:A review and evaluative comparison of state-of-the-art techniques",*PEC University of Technology,Chandigarh,India*

[26] Rishi R.Rakesh,Probal Chaudari and C.A Murthy,"Thresholding in Edge Detection:A Statistical Approach",*IEEE Transactions on Image Processing*,Vol.13,No.7,July 2004

[27] P.E Trahanias,A.N. Venetsanopoulos,"Color Edge Detection using Vector Order Statistics",*IEEE Transactions on Image Processing*,Vol.2,No.2,April 1993.

[28] Ramakant Nevatia ,"A Color Edge Detector and Its Use in Scene Segmentation",*IEEE Transactions on Systems,MAN, and Cybernetics*,Vol.SMC-7,No.11,November 1977.

[29] Genyun Sun,Xikui Sun,Xujun Han,"A New Method for Edge Detection Based on the Criterion of Separability",*Journal of Multimedia*,Vol.6,No.1,February 2011

[30] Mohammed A.El-Sayad,"A New Algorithm Based Entropic Threshold for Edge Detection in Images",*International Journal of Computer Science Issues(IJCSI)*, Vol. 8, Issue 5, No 1, September 2011