Navigation of Mobile Robots Using Artificial Intelligence Technique

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

in

Mechanical Engineering

By

Rishi Gupta & Ashish Namdeo Pendam

Department of Mechanical Engineering

National Institute of Technology

Rourkela

2007

Navigation of Mobile Robots Using Artificial Intelligence Technique

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Mechanical Engineering

By

Rishi Gupta & Ashish Namdeo Pendam

Under the guidance of:

S.K.Patel

Department of Mechanical Engineering

National Institute of Technology

Rourkela

2007

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled **“Navigation of mobile robots using artificial intelligence technique.’’** submitted by **Rishi Gupta**, **Roll No: 10303063 and Ashish Namdeo Pendam , Roll No: 10303064** in the partial fulfillment of the requirement for the degree of **Bachelor of Technology** in **Mechanical Engineering**, National Institute of Technology, Rourkela , is being carried out under my supervision.

To the best of my knowledge the matter embodied in the thesis has not been submitted to any other university/institute for the award of any degree or diploma.

Date

Professor

S.K.Patel

Department of Mechanical Engineering

National Institute of Technology

Rourkela-769008

Acknowledgment

We avail this opportunity to extend our hearty indebtedness to our guide **Professor S. K. Patel**, Mechanical Engineering Department & **Professor D.R.K.Parhi** Mechanical Engineering Department, for their valuable guidance, constant encouragement and kind help at different stages for the execution of this dissertation work.

We also express our sincere gratitude to **Dr. B. K. Nanda**, Head of the Department, Mechanical Engineering, for providing valuable departmental facilities.

Submitted by:

Rishi Gupta
Roll No: 10303063
Mechanical Engineering
National Institute of Technology
Rourkela

Ashish Namdeo Pendam
Roll No: 10303064
Mechanical Engineering
National Institute of Technology
Rourkela

CONTENTS

ABSTRACT

The ability to acquire a representation of the spatial environment and the ability to localize within it are essential for successful navigation in a-priori unknown environments. This document presents a computer vision method and related algorithms for the navigation of a robot in a static environment. Our environment is a simple white colored area with black obstacles and robot (with some identification mark-a circle and a rectangle of orange color which helps in giving it a direction) present over it .This environment is grabbed in a camera which sends image to the desktop using data cable. The image is then converted to the binary format from jpeg format using software which is then processed in the computer using MATLAB. The data acquired from the program is then used as an input for another program which controls the robot drive motors using wireless controls. Robot then tries to reach its destination avoiding obstacles in its path. .The algorithm presented in this paper uses the distance transform methodology to generate paths for the robot to execute. This paper describes an algorithm for approximately finding the fastest route for a vehicle to travel one point to a destination point in a digital plain map, avoiding obstacles along the way.

In our experimental setup the camera used is a SONY HANDYCAM. This camera grabs the image and specifies the location of the robot (starting point) in the plain and its destination point. The destination point used in our experimental setup is a table tennis ball, but it can be any other entity like a single person, a combat unit or a vehicle.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 NAVIGATION: The term navigation means the process of planning and directing the route or course of a robot or a vehicle.

1.2 ROBOT: A robot is an electro-mechanical device that can perform autonomous or preprogrammed task.

A robot may act as under the direct control of human or autonomously under the control of a programmed computer. Robots may be used to perform tasks that are dangerous or difficult for humans to implement directly (e.g. nuclear waste clean up), or may be used to automate repetitive tasks that can be performed with more precision by a robot than the employment of a human (e.g. automobile production)

Robots may be controlled directly by a human, such as remotely controlled bomb disposal robots, robotic arms, or shuttles, or may act according to their own decision making ability, provided by Artificial Intelligence. However, the majority of robots fall between these extremes, being controlled by pre-programmed computers. Such robots may include feedback loops such that they can interact with their environment, but do not display actual intelligence.

1.3. ARTIFICIAL INTELLIGENCE

It is the science of making intelligent machines, especially intelligent computer programs. It is related to similar task of using computers to understand human intelligence.

Infact, Artificial Intelligence is a branch of computer science that deals with intelligent behavior, learning and adaptation in machines.

1.4. IMAGE PROCESSING

1.4.1 Image:

An image may be defined as two dimensional function f(x, y) where x and y are spatial coordinates and amplitude of 'f' at any pair of coordinates (x, y) is called the intensity or grey level of the image at that point.

The field of digital image processing refers to processing digital images by means of a digital computer.

An image is composed of a finite no of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements, pels and pixels. Pixel is a term most widely used to denote the element of a digital image

1.5. MATLAB :

MATLAB provides a suitable environment for image processing. Although MATLAB is slower than some languages (such as C), its built in functions and syntax makes it a more versatile and faster programming environment for image processing. Once an algorithm is finalized in MATLAB, the programmer can change it to C (or another faster language) to make the program run faster..

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE REVIEW

1.Using genetic algorithms for robot motion planning

(Juan Manuel Ahuactzin, El-Ghazali Talbi, Pierre Bessiere, and Emmanuel Mazer)

Abstract:We show that the path planning problem can be expressed as an optimization problem and thus solved with a genetic algorithm. We illustrate this approach by building a path planner for a planar arm with two degree of freedom, and then we demonstrate the validity of the method by planning paths for a holonomic mobile robot.

2.Spatial Learning and Localization in Animals: A Computational Model and its Implications for Mobile Robots

(Karthik Balakrishnan, Olivier Bousquet, and Vasant Honovar)

Abstract: The ability to acquire a representation of the spatial environment and the ability to localize within it are essential for successful navigation in a-priori unknown environments .This paper briefly reviews the relevant neurobiological and cognitive data and their relation to computational models of spatial learning and localization used in mobile robots . The resulting model allows a robot to learn a place-based, metric representation of space in a-priori unknown environments and to localize itself in a stochastically optimal manner

3.Dynamic trajectory planning for cross-country navigator

(Barry L. Brumitt, R. Craig Coulter, Anthony Stentz)

Abstract:Autonomous Cross-Country Navigation requires planning algorithms which supports rapid traversal of challenging terrain while maintaining vehicle safety.The planning system uses a recursive trajectory generation algorithm, which generates spatial trajectories and then heuristically modifies them to achieve safe paths around obstacles. Velocities along the spatial trajectory are then set to ensure a dynamically stable traversal

4.An Opportunistic Global Path Planner

(John F. Canny and Ming Chieh Lin)

Abstract: In this paper we describe a robot path planning algorithm that constructs a global skeleton of free-space by incremental local methods. The curves of the skeleton are the loci of maxima of an artificial potential field that is directly proportional to [the] distance of the robot from the obstacles. Our method has the advantage of fast convergence of local methods in uncluttered environments, but it also has a deterministic and efficient method of escaping local extremal points of the potential function.

5.On All-Terrain Vehicle Motion Planning

(Moez Cherif)

Abstract: In this paper, we address the problem of motion planning for a mobile robot moving on a hilly three dimensional terrain, and subject to dynamic and physical interaction constraints.a mixed planning method based upon a two-level approach combining a discrete search strategy operating on a subset of the configuration space of the robot, and a continuous motion generation technique considering the kinematic and dynamic constraints of the task

6.Robust Path Planning in the Plane

(Fernando De la Rosa, Christian Laugier, and Jose Najera)

Abstract: This work presents an approach to plan motion strategies for robotics tasks constrainted by uncertainty in position, orientation and control. Our approach operates in a (x, y, theta) configuration space and it combines two local functions: a contact-based attraction function and an exploration function. Compliant motions are used to reduce the position/orientation uncertainty. An explicit geometric model for the uncertainty is defined to evaluate the reachability of the obstacle surfaces when the robot translates in free space.

7.The Geometrical Representation of Path Planning Problems

(Leo Dorst, Indur Mandhyan, and Karen Trovato)

Abstract: The path planning problem for arbitrary devices is first and foremost a geometrical problem. For the field of control theory, advanced mathematical techniques have been developed to describe and use geometry. In this paper, we use the notations of the flow of vector fields and geodesics in metric spaces to formalize and unify path planning problems. A path planning algorithm based on flow propagation is briefly discussed. Applications to the theory to motion planning for a robot arm, a maneuvering car, and Rubik's Cube are given. These very different problems (holonomic, non-holonomic and discrete, respectively) are solved by the same unified procedure.

8.Collision-Free Object Movement Using Vector Fields

(Parris K. Egbert and Scott H. Winkler)

Abstract:We present a technique for automatically providing animation and collision avoidance in a general-purpose computer graphics system. The technique, which relies on an expanded notion of vector fields, allows users to easily set up and animate objects, then prevents objects from colliding as the animation proceeds.

9.Finding an Unpredictable Target in a Workspace with Obstacles

(Steven LaValle, David Lin, Leonidas Guibas, Jean-Claude Latombe, and Rajeev Motwani**)**

Abstract: This paper introduces a visibility-based motion planning problem in which the task is to coordinate the motions of one or more robots that have omnidirectional vision sensors, to eventually "see" a target that is unpredictable, has unknown initial position, and is capable of moving arbitrarily fast. A visibility region is associated with each robot, and the goal is to guarantee that the target will ultimately lie in at least one visibility region . A complete algorithm for computing the motion strategy of the robots is also presented.

10.A Mobile Robot Navigation Exploration Algorithm
(Alexander Zelinsky)

Abstract: This paper will present an algorithm for path planning to a goal with a mobile robot in an unknown environment. The robot maps the environment only to the extent that is necessary to achieve the goal. Paths are generated by treating unknown regions in the environment as free space. As obstacles are encountered en route to a goal, the model of the environment is updated and a new path to the goal is planned and executed.The algorithm presented in this paper makes use of the quadtree data structure to model the environment and uses the distance transform methodology to generate paths for the robot to execute.

11.An optimal pathfinder for vehicles in real-world digital terrain maps
(Frank Markus Jönsson)
Abstract: This paper describes an algorithm for approximately finding the fastest route for a vehicle to travel between two points in a digital terrain map, avoiding obstacles along the way. The enemies are avoided by staying out of their line of sight. However, the general results of this paper should be feasable for a much wider range of applications ranging from complex GIS [Geographic Information Systems] systems to home computer games. The approach taken in this work is to translate the problem into a least cost path graph problem with an associated cost function on the graph edges

12.Smart Moves: Intelligent Pathfinding

(Bryan Stout)

Introduction: Of all the decisions involved in computer-game AI, the most common is probably pathfinding -- looking for a good route for moving an entity from here to there. The entity can be a single person, a vehicle, or a combat unit; the genre can be an action game, a simulator, a role-playing game, or a strategy game. But any game in which the computer is responsible for moving things around has to solve the pathfinding problem. And this is not a trivial problem. Questions about pathfinding are regularly seen in online game programming forums, and the entities in several games move in less than intelligent paths. However, although pathfinding is not trivial, there are some well-established, solid algorithms that deserve to be known better in the game community.

13.Shortest Paths Algorithms: Theory and Experimental Evaluation

(Boris Cherkassky, Andrew Goldberg, and Tomasz Radzik**)**

Abstract: We conduct an extensive computational study of shortest paths algorithms, including some very recent algorithms. We also suggest new algorithms motivated by the experimental results and prove interesting theoretical results suggested by the experimental data. Our computational study is based on several natural problem classes which identify strengths and weaknesses of various algorithms. These problem classes and algorithm implementations form an environment for testing the performance of shortest paths algorithms. The interaction between the experimental evaluation of algorithm behavior and the theoretical analysis of algorithm performance plays an important role in our research.

14. Shortest Path Algorithms in Transportation Models: Classical and Innovative Aspects

(Stefano Pallottino and Maria Grazia Scutella**)**

Abstract: Shortest Path Problems are among the most studied network flow optimization problems, with interesting applications in various fields. One such field is transportation, where various kinds of shortest path problems need to be solved . The aim of this work is to present in a unifying framework both the main algorithmic approaches for solving the shortest path problems that arise most frequently in the transportation field, and some important implementation techniques which allow effient procedures.

CHAPTER 3

BUILDING HARWARE

HARDWARE:

3.1. Camera: The camera used in our experimental setup is a SONY HANDYCAM.

Circuit diagram:

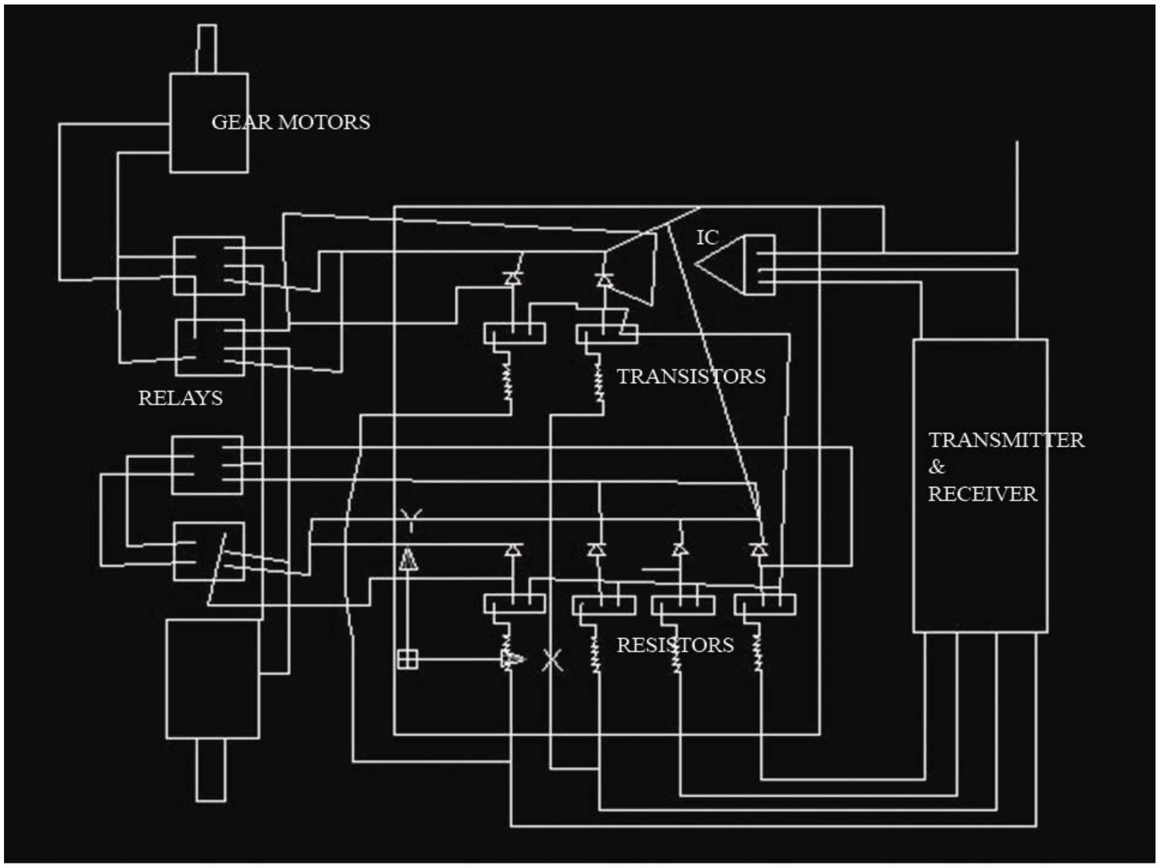


FIGURE 3.1

3.2. Components specifications and their properties:

1. Resistors:
Electrical resistance is a measure of the degree to which an object opposes the passage of an electric current.

Specification: brown black orange

2. Diode:
A diode is a component that restricts the direction of movement of charge carrier’s .Essentially it allows an electric current to flow in one direction, but blocks it in the opposite direction.

Specification: 1N 4007
Quantity: 6

3. Relays:
Relays is an automatic switch that detects conditions in circuits and causes the contacts to change their positions to close or open the circuit as required

Specification: 12V

Quantity: 4

4. Transistor:
A transistor is a semiconductor device the most common type of which acts as a current amplifier. There do however exits less common types of transistors FETS (field effect transistors) which act as voltage controlled resistors. The transistors is the fundamental building block of the circuitry that governs the operation of computers, cellular phones and all other modern electronics

Specification: BD139
Quantity: 6

5. Transmitter and receiver:
As the name suggests that it is form of pair in which a signal is transmitted and receiver receives it. The signal depends upon the frequency

Specification: 27 Hz
Quantity 1 pair

6. Geared motors:

There are the simple dc motors in which different gears are aligned to each other in order to get the required torque and RPM

Specification: 100 rpm
Quantity:2

7. Regulating IC

A voltage regulating IC is an electrical regulator designed to automatically maintain a contrast voltage level

8. **Battery**: The battery used is a Nokia mobile-2600.

Specification: 3.7 V
Quantity: 4.0

CHAPTER 4

IMAGE PROCESSING

4. DIGITAL IMAGE PROCESSING

4.1 Image:

An image may be defined as two dimensional function f(x, y) where x and y are spatial coordinates and amplitude of 'f' at any pair of coordinates (x, y) is called the intensity or grey level of the image at that point.

The field of digital image processing refers to processing digital images by means of a digital computer.

An image is composed of a finite no of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements, pels and pixels. Pixel is a term most widely used to denote the element of a digital image

4.2 Possible Representation of Images:

1. Matrix
2. Quad Trees
3. Chains } special purpose (shape, compression)
4. Pyramids

Matrix Representation is the most widely used and is 2-D array of points.

Pixel is a point on this 2-D grid. When we say pixel, we mean to say the location of the points in 2-D array, i.e., in the form of (i, j).But when we consider the pixel value, we mean to say that the grey scale value of the point or the intensity of darkness of that point, i.e. it is represented as Matrix [i] [j].

Each pixel can take one of two value, namely, 0 and 1. A pixel is either ON or OFF,

Different stages of image processing are:

1. Image acquisition
2. Image enhancement
3. Segmentation
4. Contour Following / Edge detection
5. Object recognition

4.3 Introduction to image processing:

Modern digital technology has made it possible to manipulate multi-dimensional signals with systems that range from simple digital circuits to advanced parallel computers. The goal of this manipulation can be divided into three categories:

* Image Processing *image in -> image out*

* Image Analysis *image in -> measurements out*

* Image Understanding *image in -> high-level description out*

We will focus on the fundamental concepts of *image processing*. Space does not permit us to make more than a few introductory remarks about *image analysis*. *Image understanding* requires an approach that differs fundamentally from the theme of this book. Further, we will restrict ourselves to two-dimensional (2D) image processing although most of the concepts and techniques that are to be described can be extended easily to three or more dimensions. Readers interested in either greater detail than presented here or in other aspects of image processing are referred to

We begin with certain basic definitions. An image defined in the "real world" is considered to be a function of two real variables, for example, a(x, y) with a as the amplitude (e.g. brightness) of the image at the *real* coordinate position (x, y). An image may be considered to contain sub-images sometimes referred to as *regions-of-interest*, *ROIs*, or simply *regions*. This concept reflects the fact that images frequently contain collections of objects each of which can be the basis for a region. In a sophisticated image processing system it should be possible to apply specific image processing operations to selected regions. Thus one part of an image (region) might be processed to suppress motion blur while another part might be processed to improve color rendition.

The amplitudes of a given image will almost always be either real numbers or integer numbers. The latter is usually a result of a quantization process that converts a continuous range (say, between 0 and 100%) to a discrete number of levels. In certain image-forming processes, however, the signal may involve photon counting which implies that the amplitude would be inherently quantized. In other image forming procedures, such as magnetic resonance imaging, the direct physical measurement yields a complex number in the form of a real magnitude and a real phase. For the remainder of this book we will consider amplitudes as reals or integers unless otherwise indicated.

4.4 Digital Image Definitions

A digital image $a[m,n]$ described in a 2D discrete space is derived from an analog image $a(x,y)$ in a 2D continuous space through a *sampling* process that is frequently referred to as digitization. The mathematics of that sampling process will be described in Section 5. For now we will look at some basic definitions associated with the digital image. The effect of digitization is shown in Figure 1.

The 2D continuous image $a(x,y)$ is divided into N *rows* and M *columns*. The intersection of a row and a column is termed a *pixel*. The value assigned to the integer coordinates $[m,n]$ with $\{m=0,1,2,...,M-1\}$ and $\{n=0,1,2,...,N-1\}$ is $a[m,n]$. In fact, in most cases $a(x,y)$--which we might consider to be the physical signal that impinges on the face of a 2D sensor--is actually a function of many variables including depth (z), color (λ), and time (t). Unless otherwise stated, we will consider the case of 2D, monochromatic, static images in this chapter.

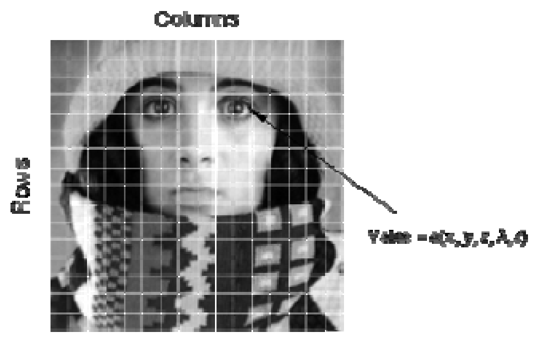


Figure 4.1: Digitization of a continuous image. The pixel at coordinates [m=10, n=3] has the integer brightness value 110.

The image shown in Figure 4.1 has been divided into N = 16 rows and M = 16 columns. The value assigned to every pixel is the average brightness in the pixel rounded to the nearest integer value. The process of representing the amplitude of the 2D signal at a given coordinate as an integer value with L different gray levels is usually referred to as amplitude quantization or simply *quantization*.

4.5 Basic Enhancement and Restoration Techniques

- Unsharp masking
- Noise suppression
- Distortion suppression

The process of image acquisition frequently leads (inadvertently) to image degradation. Due to mechanical problems, out-of-focus blur, motion, inappropriate illumination, and noise the quality of the digitized image can be inferior to the original. The goal of *enhancement* is-- starting from a recorded image $c[m,n]$--to produce the most visually pleasing image $\hat{a}[m,n]$. The goal of restoration is--starting from a recorded image $c[m,n]$--to produce the best possible estimate $\hat{a}[m,n]$ of the original image $a[m,n]$. The goal of enhancement is beauty; the goal of restoration is truth.

The measure of success in restoration is usually an error measure between the original $a[m,n]$ and the estimate $\hat{a}[m,n]$: $E\{\hat{a}[m,n], a[m,n]\}$. *No mathematical error function is known that corresponds to human perceptual assessment of error.* The mean-square error function is commonly used because:

1. It is easy to compute;

2. It is differentiable implying that a minimum can be sought;

3. It corresponds to "signal energy" in the total error, and;

4. It has nice properties *vis à vis* Parseval's theorem, eqs. (22) and (23).

The *mean-square error* is defined by:

$$E\{\hat{a},a\} = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|\hat{a}[m,n]-a[m,n]\right|^2$$

In some techniques an error measure will not be necessary; in others it will be essential for evaluation and comparative purposes.

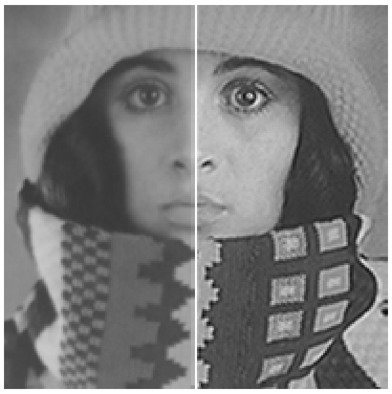
4.6 Unsharp masking

A well-known technique from photography to improve the visual quality of an image is to enhance the edges of the image. The technique is called *unsharp masking*. Edge enhancement means first isolating the edges in an image, amplifying them, and then adding them back into the image. Examination of Figure 33 shows that the Laplacian is a mechanism for isolating the gray level edges. This leads immediately to the technique:

$$\hat{a}[m,n] = a[m,n] - \left(k \bullet \nabla^2 a[m,n]\right)$$

The term k is the amplifying term and $k > 0$. The effect of this technique is shown in Figure 48.

The Laplacian used to produce Figure 4.2 is given by eq. (120) and the amplification term $k = 1$.



Original ↑↑Laplacian-enhanced

Figure 4.2 : Edge enhanced compared to original

4.7 Noise suppression

The techniques available to suppress noise can be divided into those techniques that are based on temporal information and those that are based on spatial information. By temporal information we mean that a sequence of images $\{a_p[m,n] \mid p=1,2,...,P\}$ are available that contain *exactly* the same objects and that differ only in the sense of independent noise realizations. If this is the case and if the noise is additive, then simple averaging of the sequence:

Temporal averaging $\hat{a}[m,n] = \frac{1}{P}\sum_{p=1}^{P} a_p[m,n]$ -

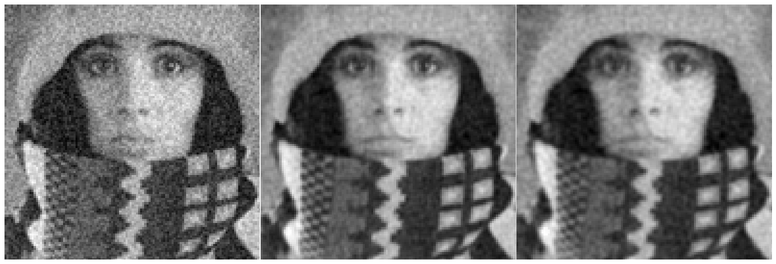
It will produce a result where the mean value of each pixel will be unchanged. For each pixel, however, the standard deviation will decrease from σ to $\sigma/\sqrt{P}$.

If temporal averaging is not possible, then spatial averaging can be used to decrease the noise. This generally occurs, however, at a cost to image sharpness. Four obvious choices for spatial averaging are the smoothing algorithms that have been described in Section 9.4 - Gaussian filtering (eq. (93)), median filtering, Kuwahara filtering, and morphological smoothing (eq.).

Within the class of linear filters, the optimal filter for restoration in the presence of noise is given by the *Wiener filter* . The word "optimal" is used here in the sense of minimum mean-square error (*mse*). Because the square root operation is monotonic increasing, the optimal filter also minimizes the root mean-square error (*rms*). The Wiener filter is characterized in the Fourier domain and for additive noise that is independent of the signal it is given by:

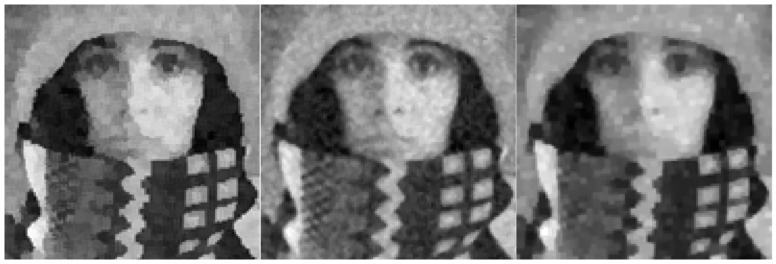
$$H_W(u,v) = \frac{S_{aa}(u,v)}{S_{aa}(u,v) + S_{nn}(u,v)}$$

where $S_{aa}(u,v)$ is the power spectral density of an ensemble of random images $\{a[m,n]\}$ and $S_{nn}(u,v)$ is the power spectral density of the random noise. If we have a single image then $S_{aa}(u,v) = |A(u,v)|^2$. In practice it is unlikely that the power spectral density of the uncontaminated image will be available. Because many images have a similar power spectral density that can be modeled which can be used as an estimate of $S_{aa}(u,v)$. A comparison of the five different techniques described above is shown in Figure 49. The Wiener filter was constructed directly from eq. because the image spectrum and the noise spectrum were known. The parameters for the other filters were determined choosing that value (either σ or window size) that led to the minimum *rms*.



a) Noisy image (*SNR*=20 *dB*) **b)** Wiener filter **c)** Gauss filter (σ = 1.0)

rms = 25.7 *rms* = 20.2 *rms* = 21.1



d) Kuwahara filter (5 x 5) **e)** Median filter (3 x 3) **f)** Morph. smoothing (3 x 3)

rms = 22.4 *rms* = 22.6 *rms* = 26.2

Figure 4.3: Noise suppression using various filtering techniques.

The root mean-square errors (*rms*) associated with the various filters are shown in Figure 4.3. For this specific comparison, the Wiener filter generates a lower error than any of the other procedures that are examined here. The two linear procedures, Wiener filtering and Gaussian filtering, performed slightly better than the three non-linear alternatives.

4.8 Distortion suppression

The model presented above--an image distorted solely by noise--is not, in general, sophisticated enough to describe the true nature of distortion in a digital image. A more realistic model includes not only the noise but also a model for the distortion induced by lenses, finite apertures, possible motion of the camera and/or an object, and so forth. One frequently used model is of an image $a[m,n]$ distorted by a linear, shift-invariant system $h_o[m,n]$ (such as a lens) and then contaminated by noise $\kappa[m,n]$. Various aspects of $h_o[m,n]$ and $\kappa[m,n]$ have been discussed in earlier sections. The most common combination of these is the additive model:

$$c[m,n] = \left(a[m,n] \otimes h_o[m,n]\right) + \kappa[m,n]$$

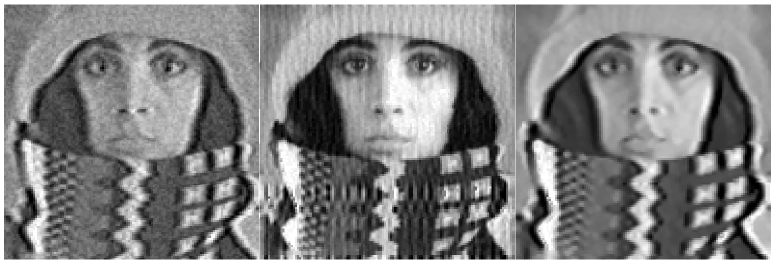
The restoration procedure that is based on linear filtering coupled to a minimum mean-square error criterion again produces a Wiener filter :

$$H_W(u,v) \quad = \frac{H_o^*(u,v) S_{aa}(u,v)}{\left|H_o(u,v)\right|^2 S_{aa}(u,v) + S_{nn}(u,v)}$$

$$= \frac{H_o^*(u,v)}{\left|H_o(u,v)\right|^2 + \left(S_{nn}(u,v) \big/ S_{aa}(u,v)\right)}$$

Once again $S_{aa}(u,v)$ is the power spectral density of an image, $S_{nn}(u,v)$ is the power spectral density of the noise, and $_o(u,v) = F\{h_o[m,n]\}$. Examination of this formula for some extreme cases can be useful. For those frequencies where $S_{aa}(u,v) >> S_{nn}(u,v)$, where the signal spectrum dominates the noise spectrum, the Wiener filter is given by $1/_o(u,v)$, the *inverse filter* solution. For those frequencies where $S_{aa}(u,v) << S_{nn}(u,v)$, where the noise spectrum dominates the signal spectrum, the Wiener filter is proportional to $_o*(u,v)$, the *matched filter* solution. For those frequencies where $_o(u,v) = 0$, the Wiener filter $_W(u,v) = 0$ preventing overflow.

The Wiener filter is a solution to the restoration problem based upon the hypothesized use of a linear filter and the minimum mean-square (or *rms*) error criterion. In the example below the image $a[m,n]$ was distorted by a bandpass filter and then white noise was added to achieve an $SNR = 30$ dB. The results are shown in Figure 4.4 .



a) Distorted, noisy image **b)** Wiener filter **c)** Median filter (3 x 3)

rms = 108.4 *rms* = 40.9 **Figure 4.4**: Noise *and* distortion suppression using the Wiener filter, eq. and the median filter.

The *rms* after Wiener filtering but before contrast stretching was 108.4; after contrast stretching with eq. (77) the final result as shown in Figure 4.4b has a mean-square error of 27.8. Using a 3 x 3 *median filter* as shown in Figure 4.4c leads to a *rms* error of 40.9 before contrast stretching and 35.1 after contrast stretching. Although the Wiener filter gives the minimum *rms* error over the set of all *linear* filters, the *non-linear* median filter gives a lower *rms* error. The operation *contrast stretching* is itself a non-linear operation. The "visual quality" of the median filtering result is comparable to the Wiener filtering result. This is due in part to periodic artifacts introduced by the linear filter which are visible in Figure 4.4 b.

4.9 Displays

The displays used for image processing--particularly the display systems used with computers--have a number of characteristics that help determine the quality of the final image.

- Refresh Rate
- Interlacing
- Resolution

4.10 Image differencing

4.10.1Image differencing is an image processing technique used to determine changes between images. The difference between two images is calculated by finding the difference between each pixel in each image, and generating an image based on the result. For this technique to work, the two images must first be aligned so that corresponding points coincide, and their photometric values must be made compatible, either by careful calibration, or by post-processing. The complexity of the pre-processing needed before differencing varies with the type of image.

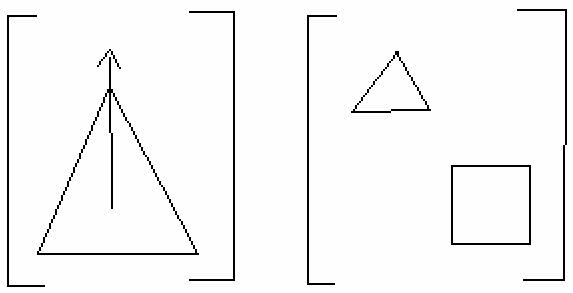
Image differencing techniques are commonly used in astronomy to locate objects that fluctuate in brightness or move against the star field.

It is the process of checking each individual pixel to see whether it belongs to an onject of interest or not.

After segmentation, it is known which pixel belongs to which object. Segmentation is also used to find the centre of an object in the image and its direction. It also differentiates between the objects in an image.

Let us consider an image having triangle only in it as shown in the figure 4.5.

Fig. 4.5



Then segmentation is used to find the centroid of the Fig (in this case triangle) and its direction as shown in the Figure 4.5. As computer is solving an image in the binary or some other form, it cannot tell what are the abjects in the object (image) and how many objects are present in the image.

Let us consider that the image being studied is having white background and the object in it are having the color which is significantly different from the background. Let the background be represtnted by 1. Such kind of files are called binary image. Now this image is shown in the figure.

$$A\,[5]\,[11] = \begin{array}{ccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

The triangle in the figure 4.5 van be represented as in array shown above.

Now we consider another array B [5] [11] as

$$B\,[5]\,[11] = \begin{array}{ccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 \\ 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 \\ 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 & 41 & 42 & 43 \\ 44 & 45 & 46 & 47 & 48 & 49 & 50 & 51 & 52 & 53 & 54 \end{array}$$

Now when we multiply A [5] [11] and B [5] [11] i.e. each element of A [i] [j] is multiplied by corresponding number in the B [i] [j], then the final matrix C [5] [11] can be written as

$$C\,[5]\,[11] = \begin{array}{ccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 26 & 27 & 28 & 0 & 0 & 0 & 0 \end{array}$$

$$\begin{matrix} 0 & 0 & 0 & 36 & 37 & 38 & 39 & 40 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Now if a square is also present in the image then the matrix can be represented a shown.

$$C\,[5]\,[11] = \begin{matrix} 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 12 & 13 & 14 & 0 & 0 & 0 & 18 & 19 & 20 & 0 \\ 22 & 23 & 24 & 25 & 26 & 0 & 0 & 29 & 30 & 31 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 40 & 41 & 42 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

4.10.2 Algorithm

In this Section we will describe operations that are fundamental to digital image processing. These operations can be divided into four categories: operations based on the image histogram, on simple mathematics, on convolution, and on mathematical morphology. Further, these operations can also be described in terms of their implementation as a point operation, a local operation, or a global operation as described

A is our image array, as explained above

```
B [ i ] [ j ] = i * n + j                0<= i <= m-1,   0 <= j <= n
        let D [ i ] [ j ] = 0
        C = A * B
while   D != C
        D = C
        C = ( b V N ). a
end
```

This algorithm will start from the right bottom of the array and a van ~ Newman region is considered. We compare each pixel with its environment pixel and assign the maximum value in the region around it and then multiply this matrix with the original image array A. We will find the final array, that will be like shown in the figure.

$$X[5][11] = \begin{matrix} 0 & 0 & 26 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 26 & 26 & 26 & 0 & 0 & 0 & 42 & 42 & 42 & 0 \\ 26 & 26 & 26 & 26 & 26 & 0 & 0 & 42 & 42 & 42 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 42 & 42 & 42 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Now what we see in this figure is that one object is represented by an no. 26 and another object is represented by a no. 42. Now consider there is another object which would have been represented by 7, is having only one value in the figure. Then we can remove this 7 from figure by considering a threshold, i.e. minimum no. of numbers, which should be present to consider it as an object. This process is called noise removal. Now as 26 and 42 are shown in figure, they also help in finding edges.

The method of segmentation can hence be used to find the edges.

4.11 Image acquisition

Image aquisition is done by using camera, scanner or other devices.

4.12 Cameras

The cameras and recording media available for modern digital image processing applications are changing at a significant pace. To dwell too long in this section on one major type of camera, such as the CCD camera, and to ignore developments in areas such as charge injection device (CID) cameras and CMOS cameras is to run the risk of obsolescence. Nevertheless, the techniques that are used to characterize the CCD camera remain "universal" and the presentation that follows is given in the context of modern CCD technology for purposes of illustration.

The camera used in our project model is SONY HANDYCAM.

4.13 Image stabilization

Image stabilization is a family of techniques to increase the stability of an image. It is used in image-stabilized binoculars, photography, videography, and astronomical telescopes. With still cameras, camera shake is particularly problematic at slow shutter speeds or with long focal length (telephoto) lenses. With video cameras, camera shake causes visible frame-to-frame jittering in the video recorded. With astronomy, these problems are compounded by variations in the atmosphere over time, which causes the apparent position of objects to move.

4.14 Noise:

Images acquired through modern sensors may be contaminated by a variety of noise sources. By noise we refer to stochastic variations as opposed to deterministic distortions such as shading or lack of focus. We will assume for this section that we are dealing with images formed from light using modern electro-optics. In particular we will assume the use of modern, charge-coupled device (CCD) cameras where photons produce electrons that are commonly referred to as photoelectrons. Nevertheless, most of the observations we shall make about noise and its various sources hold equally well for other imaging modalities.

While modern technology has made it possible to reduce the noise levels associated with various electro-optical devices to almost negligible levels, one noise source can never be eliminated and thus forms the limiting case when all other noise sources are "eliminated".

- Photon Noise
- Thermal Noise
- On-chip Electronic Noise
- KTC Noise
- Amplifier Noise
- Quantization Noise

4.15 Image Sampling :

Converting from a continuous image a(x,y) to its digital representation b[m,n] requires the process of sampling. In the ideal sampling system a(x,y) is multiplied by an ideal 2D impulse train:

$$\begin{aligned} b_{ideal}[m.n] &= a(x,y) \bullet \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} \delta(x - mX_o, y - nY_o) \\ &= \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} a(mX_o, nY_o)\delta(x - mX_o, y - nY_o) \end{aligned}$$

where Xo and Yo are the sampling distances or intervals, d(*,*) is the ideal impulse function, and we have used eq. . (At some point, of course, the impulse function d(x,y) is converted to the discrete impulse function d[m,n].) Square sampling implies that Xo =Yo. Sampling with an impulse function corresponds to sampling with an infinitesimally small point. This, however, does not correspond to the usual situation as illustrated in Figure 1. To take the effects of a finite sampling aperture p(x,y) into account, we can modify the sampling model as follows:

$$b[m,n] = \left(a(x,y) \otimes p(x,y)\right) \bullet \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} \delta(x - mX_o, y - nY_o)$$

The combined effect of the aperture and sampling are best understood by examining the Fourier domain representation.

$$B(\Omega, \Psi) = \frac{1}{4\pi^2} \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} A(\Omega - m\Omega_s, \Psi - n\Psi_s) \bullet P(\Omega - m\Omega_s, \Psi - n\Psi_s)$$

where Ωs = 2π/Xo is the sampling frequency in the x direction and Ψs = 2π/Yo is the sampling frequency in the y direction. The aperture p(x,y) is frequently square, circular, or Gaussian with the associated P(Ω,Ψ). (See Table 4.) The periodic nature of the spectrum, described in eq. is clear from eq.

Sampling Density for Image Processing

Sampling Density for Image Analysis

4.16 Segmentation

- Thresholding
- Edge finding
- Binary mathematical morphology
- Gray-value mathematical morphology

In the analysis of the objects in images it is essential that we can distinguish between the objects of interest and "the rest." This latter group is also referred to as the background. The techniques that are used to find the objects of interest are usually referred to as segmentation techniques - segmenting the foreground from background. In this section we will two of the most common techniques--thresholding and edge finding-- and we will present techniques for improving the quality of the segmentation result. It is important to understand that:

* there is no universally applicable segmentation technique that will work for all images, and,

* no segmentation technique is perfect.

4.16.1 Thresholding:

This technique is based upon a simple concept. A parameter θcalled the brightness threshold is chosen and applied to the image a[m,n] as follows:

$$\begin{array}{ll} \textbf{If} \quad a[m,n] \geq \theta & a[m,n] = object \;= 1 \\ \textbf{Else} & a[m,n] = background \;= 0 \end{array}$$

This version of the algorithm assumes that we are interested in light objects on a dark background. For dark objects on a light background we would use:

$$\begin{array}{ll} \textbf{If} \quad a[m,n] < \theta & a[m,n] = object \;= 1 \\ \textbf{Else} & a[m,n] = background \;= 0 \end{array}$$

The output is the label "object" or "background" which, due to its dichotomous nature, can be represented as a Boolean variable "1" or "0". In principle, the test condition could be based upon some other property than simple brightness (for example, If (Redness{a[m,n]} >= θred), but the concept is clear.

The central question in thresholding then becomes: ow do we choose the threshold θ? While there is no universal procedure for threshold selection that is guaranteed to work on all images, there are a variety of alternatives.

* ***Fixed threshold*** - One alternative is to use a threshold that is chosen independently of the image data. If it is known that one is dealing with very high-contrast images where the objects are very dark and the background is homogeneous (Section 10.1) and very light, then a constant threshold of 128 on a scale of 0 to 255 might be sufficiently accurate. By accuracy we mean that the number of falsely-classified pixels should be kept to a minimum.

* ***Istogram-derived thresholds*** - In most cases the threshold is chosen from the brightness histogram of the region or image that we wish to segment

4.16.2 Edge Finding (CANNY Edge Detection):

The binary image is one of the most important branches of digital image processing, for example, X-ray images in industrial and medical images. Edge detection is a useful low-level image processing for obtaining a simplified image [1]. Numerous new techniques and complex methods have been developed in this area [2], [3]. The typical edge detection technologies generally could be classified into two types: direction derivative operations and isotropic edge operations. The isotropic edge detection gained mostly popular because it is more similar to the humans' vision mechanism [1]. The isotropic edge detectors have two different types: gradient-based operators such as Roberts, Sobel and Laplacian [4], [5], and second derivative operator proposed by Marr and Hildreth [6]. Canny method is one of the most effective edge detection methods [7], but it is difficult to implement for monochrome images [8]. While these widely known methods could obtain reasonable edge maps for most images, they were mostly based on continuous functions; this would cause distortion in detail for the non-continuous binary image.

Generally speaking, the aim of edge detection on the binary images is to classify the pixels into two opposite classes: edge versus non-edge. If the *inner pixels* of the images were directly cleared, then the *edges* can be obtained. The key problem is how to construct an operator to detect the inner pixels of the images. The Connectivity-Number (CN) of the black pixels is one of such operators. The approach proposed in this paper follows the theory developed in [9] and was called *Connectivity-Number-based Edge Detection* method. CNED belongs to isotropic edge detection technology

4.16.2.1 The principle of CNED.

In binary images, the connectivity number of the black pixel is the number of the connected black pixels passed when it moves around its neighbors [9]. It can be defined as in formula .

$$CN_n = \sum_{i \in \{0,2,4,6\}} [f(x_i) - f(x_i) * f(x_{i+1}) * f(x_{i+2})] \quad (1)$$

Here n is 4 or 8, and. $X_8 = x_0, x_1, \ldots, x_7$ is the 8-neighbors of the center pixel.
When n equals to 8, $f(x_i) = f(x_i)$.

There are totally 256 8-neighbors of the center black pixel. Considering the symmetry of the patterns, they can be classified into 48 combinations according to the number and position of the black pixels around its center pixel, as shown in Figure 4.6.

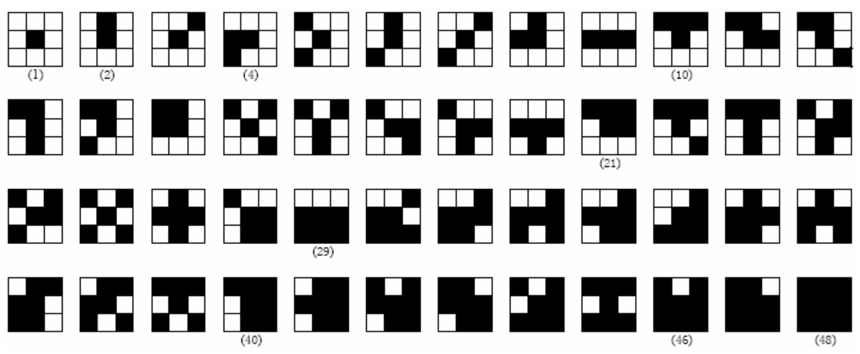


Figure 4.6: The typical 48 8-neighbors

The connectivity numbers of the above 8-neighbors were listed in Table 1.

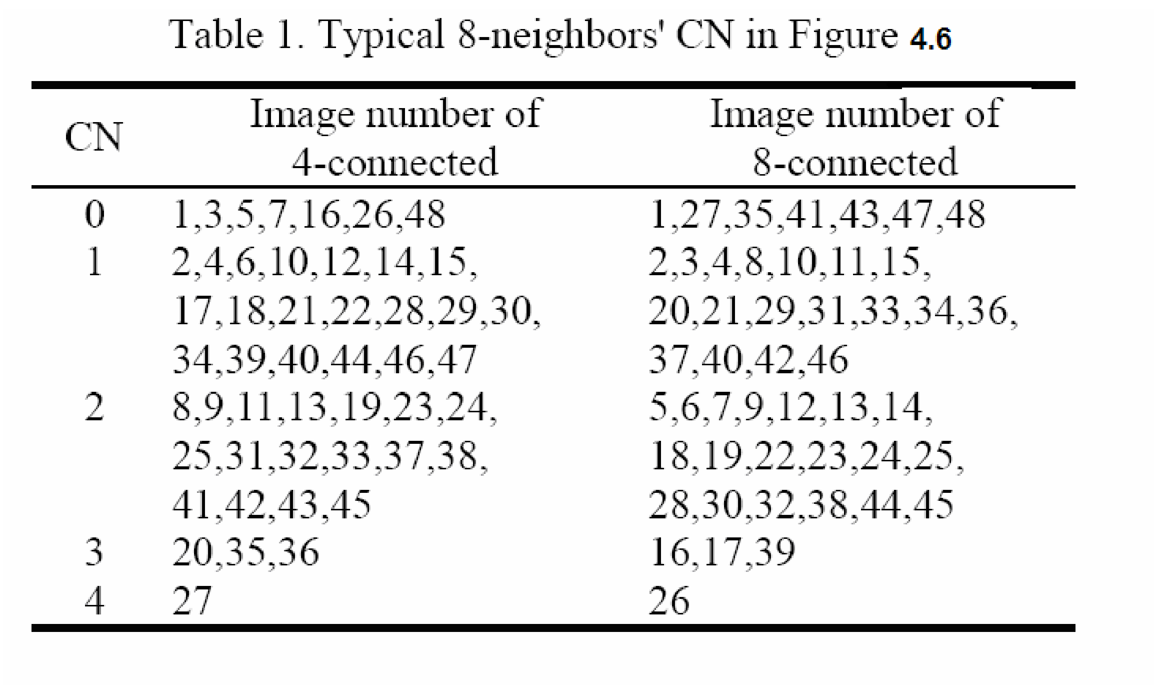


Table 1. Typical 8-neighbors' CN in Figure 4.6

CN	Image number of 4-connected	Image number of 8-connected
0	1,3,5,7,16,26,48	1,27,35,41,43,47,48
1	2,4,6,10,12,14,15, 17,18,21,22,28,29,30, 34,39,40,44,46,47	2,3,4,8,10,11,15, 20,21,29,31,33,34,36, 37,40,42,46
2	8,9,11,13,19,23,24, 25,31,32,33,37,38, 41,42,43,45	5,6,7,9,12,13,14, 18,19,22,23,24,25, 28,30,32,38,44,45
3	20,35,36	16,17,39
4	27	26

From Table 1, one can divide the 256 8-neighbors according to the pixel's CN [10], whatever it is 4- or 8-connected (Table 2.).

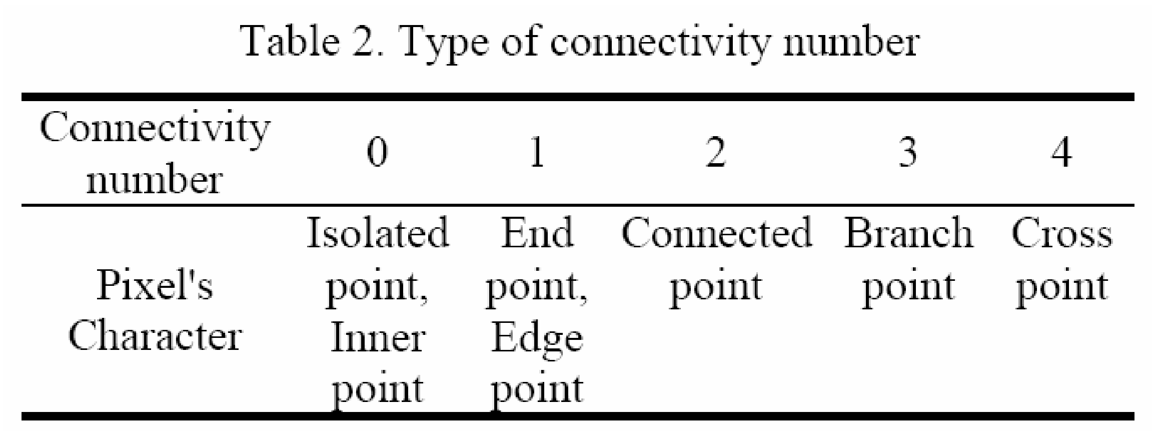


Table 2. Type of connectivity number

Connectivity number	0	1	2	3	4
Pixel's Character	Isolated point, Inner point	End point, Edge point	Connected point	Branch point	Cross point

The purpose of the CNED method is to eliminate the inner pixels. In order to improve the operation's efficiency, one can select those pixels that have zero-CN-neighbor as edge detector (here, the isolated points were also eliminated), and then apply *exclusive or* operation with the initial binary image, pixel by pixel. Thus the operation is simplified to efficient binary *exclusive or* operation, improves the operation speed. From Table 1, one also found that there are only 7 typical zero-CN 8-neighbors, while the 4-connected pixels are somewhat different from 8-connected ones. The 8-connected pixels are shown in Figure 4.7. (b) ~ (h), and the 4-connected shown in Figure 4.7(i) ~ (o).

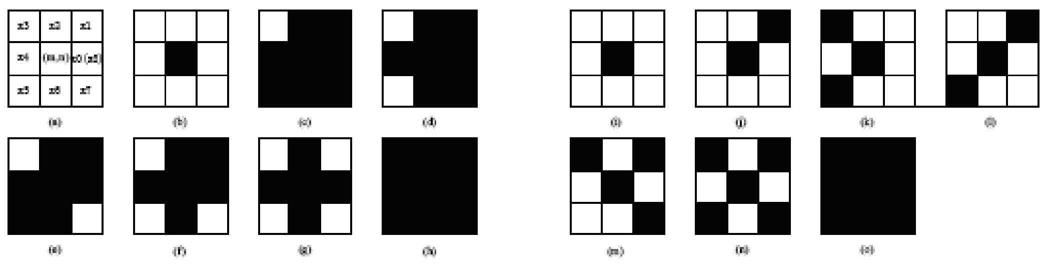


FIGURE 4.7 THE PIXEL'S NEIGHBOUR

From the above figure, one can found that in 8-connected neighbors, those 4 diagonal corners have white pixels were considered as zero-CN pixels besides the all-white and all-black neighbors; the 4-connected neighbors were just the inverse to the 8-connected, those black pixels in the 4 corners were zero-CN pixels except the center black pixels. For the 8-connected neighbors have more similarities than the 4-connedted ones with the human been, it has better visual performance. Without losing generality, only the 8-connected neighbors were discussed here. The above 7 typical zero-CN 8-connected neighbors have 17 different combination forms. When combine their neighbors from row to columns as bytes, one can found it is very regular (Table 3).

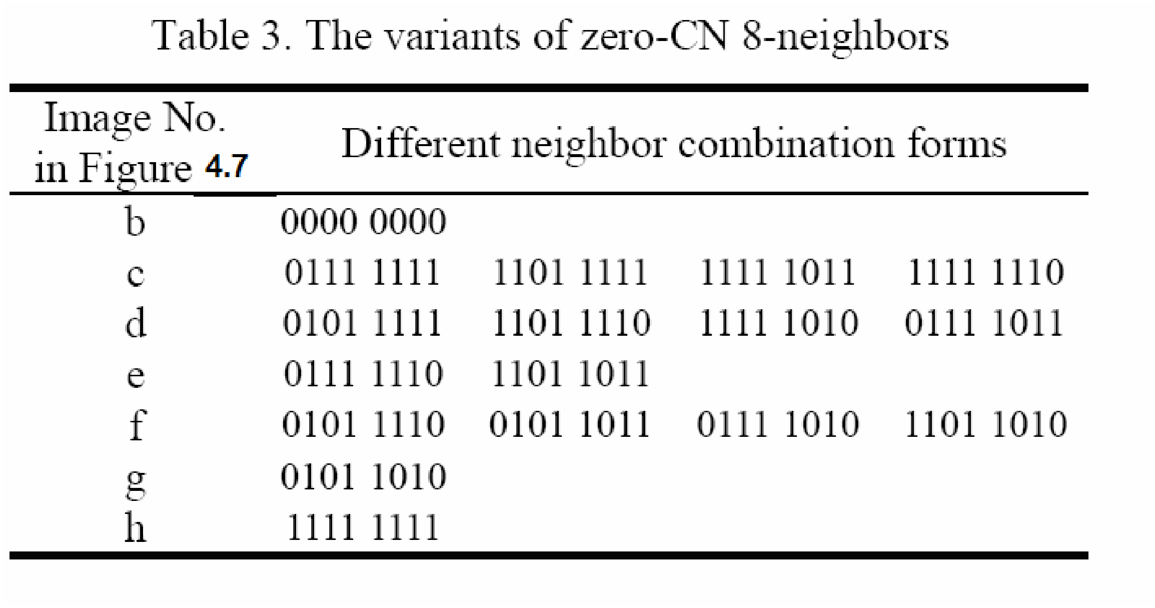


Table 3. The variants of zero-CN 8-neighbors

Image No. in Figure 4.7	Different neighbor combination forms			
b	0000 0000			
c	0111 1111	1101 1111	1111 1011	1111 1110
d	0101 1111	1101 1110	1111 1010	0111 1011
e	0111 1110	1101 1011		
f	0101 1110	0101 1011	0111 1010	1101 1010
g	0101 1010			
h	1111 1111			

From the hexadecimal format of above binary code, one gets clearer on their variance

Edge Detection can also be done using canny method.

4.17. Techniques

The algorithms presented can be used to build techniques to solve specific image processing problems. Without presuming to present the solution to all processing problems, the following examples are of general interest and can be used as models for solving related problems.

- Shading Correction
- Basic Enhancement and Restoration Techniques
- Segmentation

4.18 Tools

Certain tools are central to the processing of digital images. These include mathematical tools such as *convolution*, *Fourier analysis*, and *statistical* descriptions, and manipulative tools such as *chain codes* and *run codes*. We will present these tools without any specific motivation. The motivation will follow in later sections.

- Convolution

- Properties of Convolution
- Fourier Transforms
- Properties of Fourier Transforms
- Statistics
- Contour Representations

CHAPTER 5

BUILDING SOFTWARE

SOFTWARE:

5.1. MATLAB

MATLAB provides a suitable environment for image processing. Although MATLAB is slower than some languages (such as C), its built in functions and syntax makes it a more versatile and faster programming environment for image processing. Once an algorithm is finalized in MATLAB, the programmer can change it to C (or another faster language) to make the program run faster.

MATLAB does not have the easy to use interfaces of Adobe Photoshop. MATLAB is used to test and tweak new image processing techniques and algorithms. Almost everything in MATLAB is done through programming and manipulation of raw image data and not a user interface. The effects and filters in Photoshop (or any other image editing software) are actually algorithms. With MATLAB, the user can create these complex algorithms that are applied in Photoshop.

5.2. INTRODUCTION TO IMAGE PROCESSING IN MATLAB 1

5.2.1 Introduction

This worksheet is an introduction on how to handle images in Matlab. When working with images in Matlab, there are many things to keep in mind such as loading an image, using the right format, saving the data as different data types, how to display an image, conversion between different image formats, etc. This worksheet presents some of the commands designed for these operations. Most of these commands require you to have the Image processing tool box installed with Matlab. To find out if it is installed, type ver at the Matlab prompt. This gives you a list of what tool boxes that are installed on your system.

For further reference on image handling in Matlab you are recommended to use Matlab's help browser. There is an extensive (and quite good) on-line manual for the Image processing tool box that you can access via Matlab's help browser.

The first sections of this worksheet are quite heavy. The only way to understand how the presented commands work, is to carefully work through the examples given at the end of the worksheet. Once you can get these examples to work, experiment on your own using your favorite image!

5.2.2. Fundamentals

A digital image is composed of pixels which can be thought of as small dots on the screen. A digital image is an instruction of how to color each pixel. We will see in detail later on how this is done in practice. A typical size of an image is 512-by-512 pixels. Later on in the course you will see that it is convenient to let the dimensions of the image to be a power of 2. For example, 29=512. In the general case we say that an image is of size m-by-n if it is composed of m pixels in the vertical direction and n pixels in the horizontal direction.

Let us say that we have an image on the format 512-by-1024 pixels. This means that the data for the image must contain information about 524288 pixels, which requires a lot of memory! Hence, compressing images is essential for efficient image processing. You will later on see how Fourier analysis and Wavelet analysis can help us to compress an image significantly. There are also a few "computer scientific" tricks (for example entropy coding) to reduce the amount of data required to store an image.

Image formats supported by Matlab

The following image formats are supported by Matlab:

- BMP
- HDF
- JPEG
- PCX
- TIFF
- XWB

Most images you find on the Internet are JPEG-images which is the name for one of the most widely used compression standards for images. If you have stored an image you can usually see from the suffix what format it is stored in. For example, an image named myimage.jpg is stored in the JPEG format and we will see later on that we can load an image of this format into Matlab.

5.3 Working formats in Matlab

If an image is stored as a JPEG-image on your disc we first read it into Matlab. However, in order to start working with an image, for example perform a wavelet transform on the image, we must convert it into a different format. This section explains four common formats.

Intensity image (gray scale image)

This is the equivalent to a "gray scale image" and this is the image we will mostly work with in this course. It represents an image as a matrix where every element has a value corresponding to how bright/dark the pixel at the corresponding position should be colored. There are two ways to represent the number that represents the brightness of the pixel: The double class (or data type). This assigns a floating number ("a number with decimals") between 0 and 1 to each pixel. The value 0 corresponds to black and the value 1 corresponds to white. The other class is called uint8 which assigns an integer between 0 and 255 to represent the brightness of a pixel. The value 0 corresponds to black and 255 to white. The class uint8 only requires roughly 1/8 of the storage compared to the class double. On the other hand, many mathematical functions can only be applied to the double class. We will see later how to convert between double and uint8.

5.4. Binary image

This image format also stores an image as a matrix but can only color a pixel black or white (and nothing in between). It assigns a 0 for black and a 1 for white.

Indexed image

This is a practical way of representing color images. (In this course we will mostly work with gray scale images but once you have learned how to work with a gray scale image you will also know the principle how to work with color images.) An indexed image stores an image as two matrices. The first matrix has the same size as the image and one number for each pixel. The second matrix is called the color map and its size may be different from the image. The numbers in the first matrix is an instruction of what number to use in the color map matrix.

5.5 RGB image

This is another format for color images. It represents an image with three matrices of sizes matching the image format. Each matrix corresponds to one of the colors red, green or blue and gives an instruction of how much of each of these colors a certain pixel should use.

5.6 Multiframe image

In some applications we want to study a sequence of images. This is very common in biological and medical imaging where you might study a sequence of slices of a cell. For these cases, the multiframe format is a convenient way of working with a sequence of images. In case you choose to work with biological imaging later on in this course, you may use this format.

How to convert between different formats

The following table shows how to convert between the different formats given above. All these commands require the Image processing tool box!

5.7 Image format conversion

(Within the parenthesis you type the name of the image you wish to convert.)

Operation:	**Matlab command:**
Convert between intensity/indexed/RGB format to binary format.	dither()
Convert between intensity format to indexed format.	gray2ind()
Convert between indexed format to intensity format.	ind2gray()
Convert between indexed format to RGB format.	ind2rgb()
Convert a regular matrix to intensity format by scaling.	mat2gray()
Convert between RGB format to intensity format.	rgb2gray()
Convert between RGB format to indexed format.	rgb2ind()

The command mat2gray is useful if you have a matrix representing an image but the values representing the gray scale range between, let's say, 0 and 1000. The command mat2gray automatically re scales all entries so that they fall within 0 and 255 (if you use the uint8 class) or 0 and 1 (if you use the double class).

How to convert between double and uint8

When you store an image, you should store it as a uint8 image since this requires far less memory than double. When you are processing an image (that is performing mathematical operations on an image) you should convert it into a double. Converting back and forth between these classes is easy.

I=im2double(I);

converts an image named I from uint8 to double.

I=im2uint8(I);

converts an image named I from double to uint8.

How to read files ?

When you encounter an image you want to work with, it is usually in form of a file (for example, if you down load an image from the web, it is usually stored as a JPEG-file). Once we are done processing an image, we may want to write it back to a JPEG-file so that we can, for example, post the processed image on the web. This is done using the imread and imwrite commands. These commands require the Image processing tool box!

Reading and writing image files

Operation:	**Matlab command:**
Read an image. (Within the parenthesis you type the name of the image file you wish to read. Put the file name within single quotes ' '.)	imread()
Write an image to a file. (As the first argument within the parenthesis you type the name of the image you have worked with. As a second argument within the parenthesis you type the name of the file and format that you want to write the image to. Put the file name within single quotes ' '.)	imwrite(,)

Make sure to use semi-colon ; after these commands, otherwise you will get LOTS OF number scrolling on you screen... The commands imread and imwrite support the formats given in the section "Image formats supported by Matlab" above.

Loading and saving variables in Matlab

This section explains how to load and save variables in Matlab. Once you have read a file, you probably convert it into an intensity image (a matrix) and work with this matrix. Once you are done you may want to save the matrix representing the image in order to continue to work with this matrix at another time. This is easily done using the commands save and load. Note that save and load are commonly used Matlab commands, and works independently of what tool boxes that are installed.

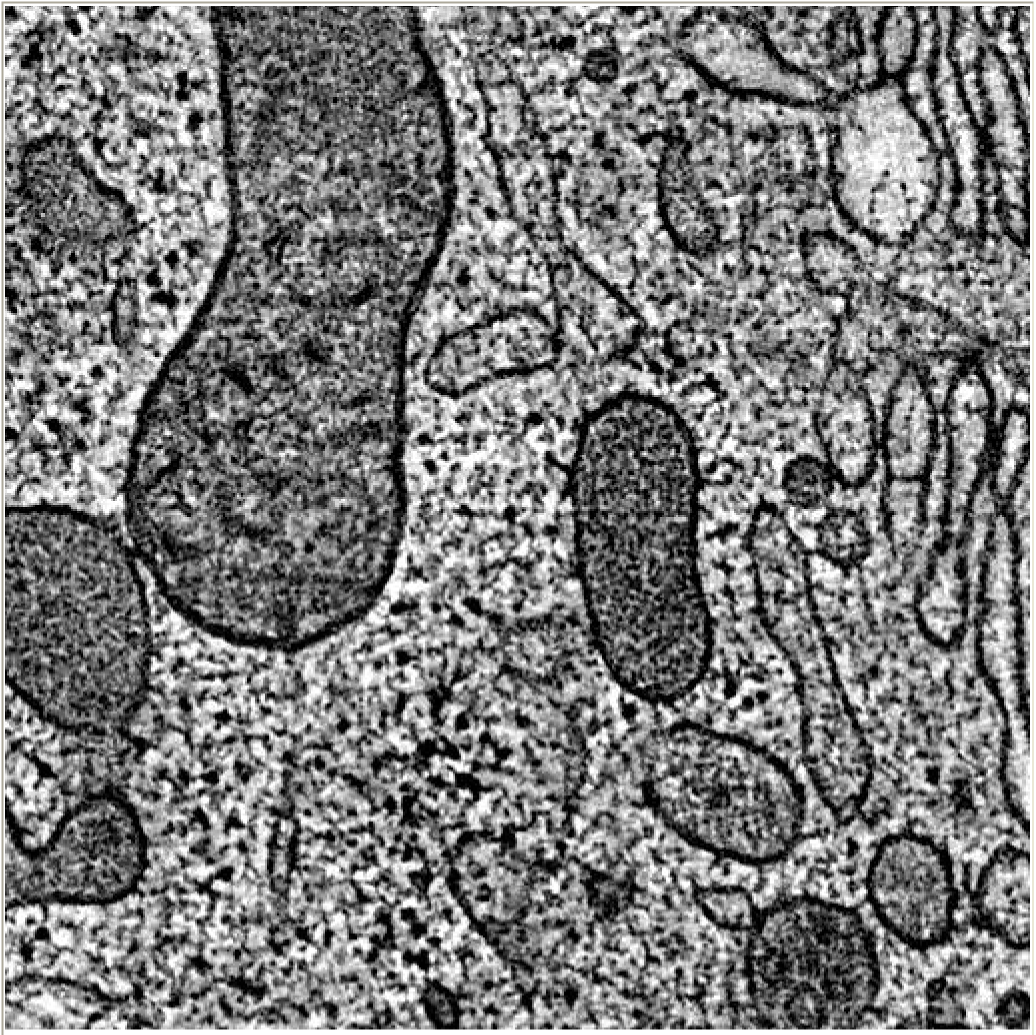
Loading and saving variables

Operation:	**Matlab command:**
Save the variable X .	save X
Load the variable X .	load X

Examples

In the first example we will down load an image from the web, read it into Matlab, investigate its format and save the matrix representing the image.

Example 5.1.



Down load the following image (by clicking on the image using the right mouse button) and save the file as cell1.jpg.
This is an image of a cell taken by an electron microscope at the Department of Molecular, Cellular and Developmental Biology at CU.

Now open Matlab and make sure you are in the same directory as your stored file. (You can check what files your directory contains by typing less at the Matlab prompt. You change directory using the command cd.) Now type in the following commands and see what each command does. (Of course, you do not have to type in the comments given in the code after the % signs.)

```
I=imread('cell1.jpg'); % Load the image file and store it as the variable I.

whos % Type "whos" in order to find out the size and class of all stored variables.

save I % Save the variable I.

ls % List the files in your directory.

% There should now be a file named "I.mat" in you directory
% containing your variable I.
```

Note that all variables that you save in Matlab usually get the suffix .mat.
Next we will see that we can display an image using the command imshow. This command requires the image processing tool box. Commands for displaying images will be explained in more detail in the section "How to display images in Matlab" below.

```
clear % Clear Matlab's memory.
```

```
load I % Load the variable I that we saved above.

whos % Check that it was indeed loaded.

imshow(I) % Display the image

I=im2double(I); % Convert the variable into double.

whos % Check that the variable indeed was converted into double

% The next procedure cuts out the upper left corner of the image
% and stores the reduced image as Ired.

for i=1:256
for j=1:256
Ired(i,j)=I(i,j);
end
end

whos % Check what variables you now have stored.

imshow(Ired) % Display the reduced image.
```

Example 5.2

Go to the CU home page and down load the image of campus with the Rockies in the background. Save the image as pic-home.jpg

Next, do the following in Matlab. (Make sure you are in the same directory as your image file).

```
clear
```

```
A=imread('pic-home.jpg');

whos

imshow(A)
```

Note that when you typed whos it probably said that the size was 300x504x3. This means that the image was loaded as an RGB image (see the section "RGB image above"). However, in this course we will mostly work with gray scale images, so let us convert it into a gray scale (or "intensity") image.

```
A=rgb2gray(A); % Convert to gray scale

whos

imshow(A)
```

Now the size indicates that our image is nothing else than a regular matrix.

Note: In other cases when you down load a color image and type whos you might see that there is one matrix corresponding to the image size and one matrix called map stored in Matlab. In that case, you have loaded an indexed image (see section above). In order to convert the indexed image into an intensity (gray scale) image, use the ind2gray command described in the section "How to convert between different formats" above.

How to display an image in Matlab

Here are a couple of basic Matlab commands (do not require any tool box) for displaying an image.

Displaying an image given on matrix form

Operation:	Matlab command:

Display an image represented as the matrix X.	imagesc(X)
Adjust the brightness. s is a parameter such that -1<s<0 gives a darker image, 0<s<1 gives a brighter image.	brighten(s)
Change the colors to gray.	colormap(gray)

Sometimes your image may not be displayed in gray scale even though you might have converted it into a gray scale image. You can then use the command colormap(gray) to "force" Matlab to use a gray scale when displaying an image.
If you are using Matlab with an Image processing tool box installed, I recommend you to use the command imshow to display an image.
Displaying an image given on matrix form (with image processing tool box)

Operation:	**Matlab command:**
Display an image represented as the matrix X.	imshow(X)
Zoom in (using the left and right mouse button).	zoom on
Turn off the zoom function.	zoom off

5.8 Image Matrices

MATLAB handles images as matrices. This involves breaking each pixel of an image down into the elements of a matrix. MATLAB distinguishes between color and grayscale images and therefore their resulting image matrices differ slightly.

5.9 Pixel values

MATLAB, by default, will use integer values (which MATLAB terms as **uint8**) that have a range of integers from 0 to 255 to represent a pixel value. 0 stands for the lightest color and 255 stands for the darkest color possible. This applies to the RGB (Red Green Blue) color channels. In the case of the Red channel, lower numbers will produce lighter (pink) values for red, and higher numbers near 255 will produce darker (maroon) values of red.

For the intensity matrix (m by n by 1) this scale applies for colors between white and black (or depends on the colormap being used).

The second type of pixel values used for images are called **double** which are floating point (decimal) numbers between 0 and 1. This range is proportional to the uint8 range and therefore multiplying each double pixel value by 255 will yield a uint8 pixel value. Similarly conversion from uint8 to double is done by dividing the uint8 value by 255.

5.10 Image Processing Toolbox

MATLAB does have extensions which contain functions geared towards image processing. They are grouped under the ‘Image Processing Toolbox’. In some versions it is an optional extra which users have to pay for, while in others it comes packaged with the software. This toolbox is especially helpful for applying numerous filters (such as linear and deblur) and also includes algorithms which can detect lines and edges in an image. A programmer can write almost all the features in this extension. Otherwise the command ‘edit commandName” usually allows a user to see/modify lines of the built-in functions that MATLAB and its Image Processing Toolbox provide.

5.11. Matlab Program:

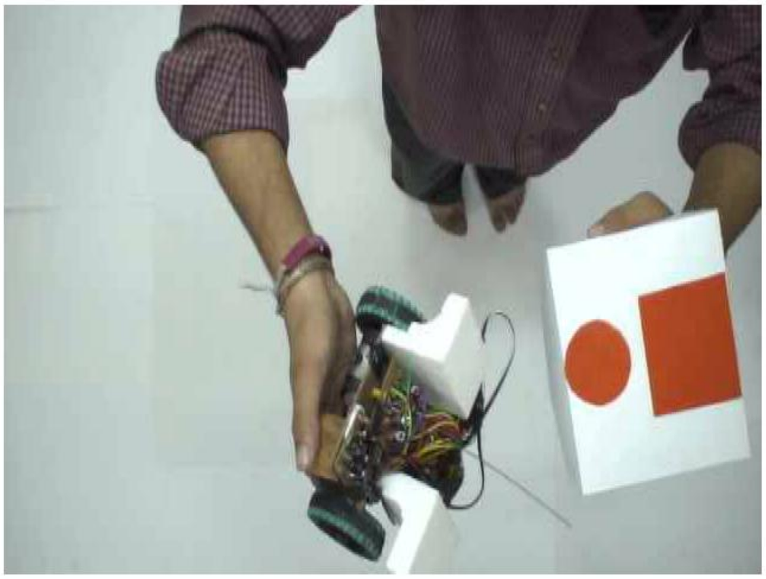
```
clear all
close all
pt=imread('arena4.jpg');
l=graythresh(pt);
ptb=im2bw(pt,l);
a=ones(50);
ptinv=a-ptb;%binary matrix background black,implements invert color
count=0;
   for i=1:50
      for j=1:50
         D(i,j)=count;
         count=count+1;
```

```
        end
    end
D ;    %serial matrix with elements ordered serially[1 2 3 ....
C=ptinv.*D;%matrix with number values at object points
C1=pad202(C,50);%pads with zeros
ptinv1=pad202(ptinv,50);
for y=51:-1:2%this block implements algo for uniformisation
    for x=51:-1:2
        c=[C1(y,x-1),C1(y-1,x),C1(y,x+1),C1(y+1,x),C1(y-1,x-1),C1(y-1,x+1),C1(y+1,x-
1),C1(y+1,x+1)];
        m=max(c);%max value in the neihbourhood
        if(m>C1(y,x))
            C1(y,x)=m;
        end
        C1=ptinv1.*C1;%C1 matrix is the uniform numbered the output matrix%
    end
end
count=0;
N=1024*1024;
A=-32767.*ones(1,N);
for i=52:-1:1
    for j=52:-1:1
        if C1(i,j)~=0
            p=1;
            while p<=N && A(p)~=C1(i,j) && A(p)~=-32767
                p=p+1;
            end

            if p<N
                if A(p)==-32767
                    A(p)=C1(i,j);
                end
            end
        end
    end
end

p=1;
while p<=N && A(p)~=-32767
    x(p)=A(p);
    count=count+1;
    p=p+1;
end

x
sprintf('Count is %d',count)
   cen=zeros(count,2);%centroid matrix
   for n=1:count
    [cen(n,1) cen(n,2)]=centlocat1(C1,52,x(n));% centroid output stored here
    End
```

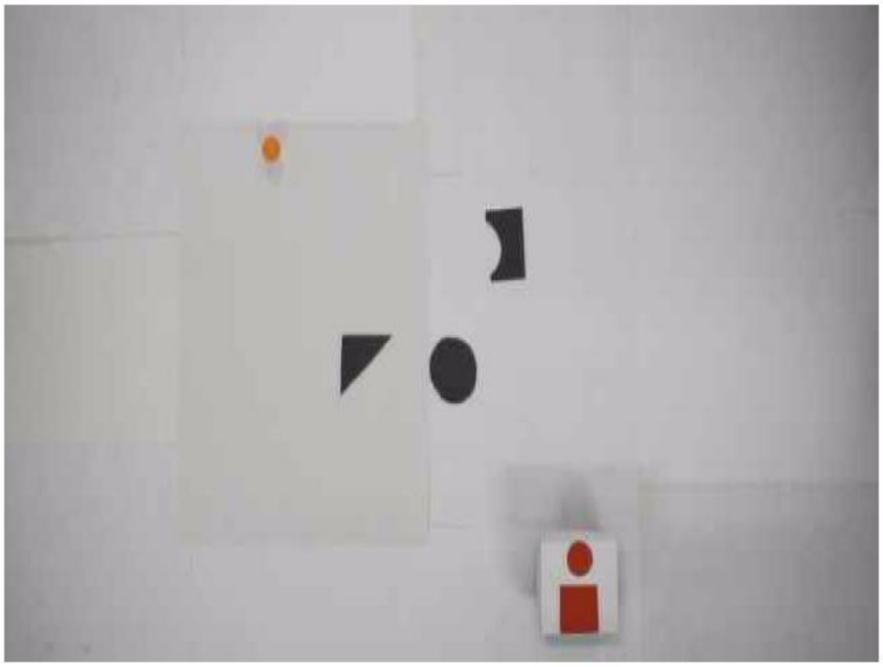
CHAPTER 6

FINAL DESIGN

ROBOT



ARENA

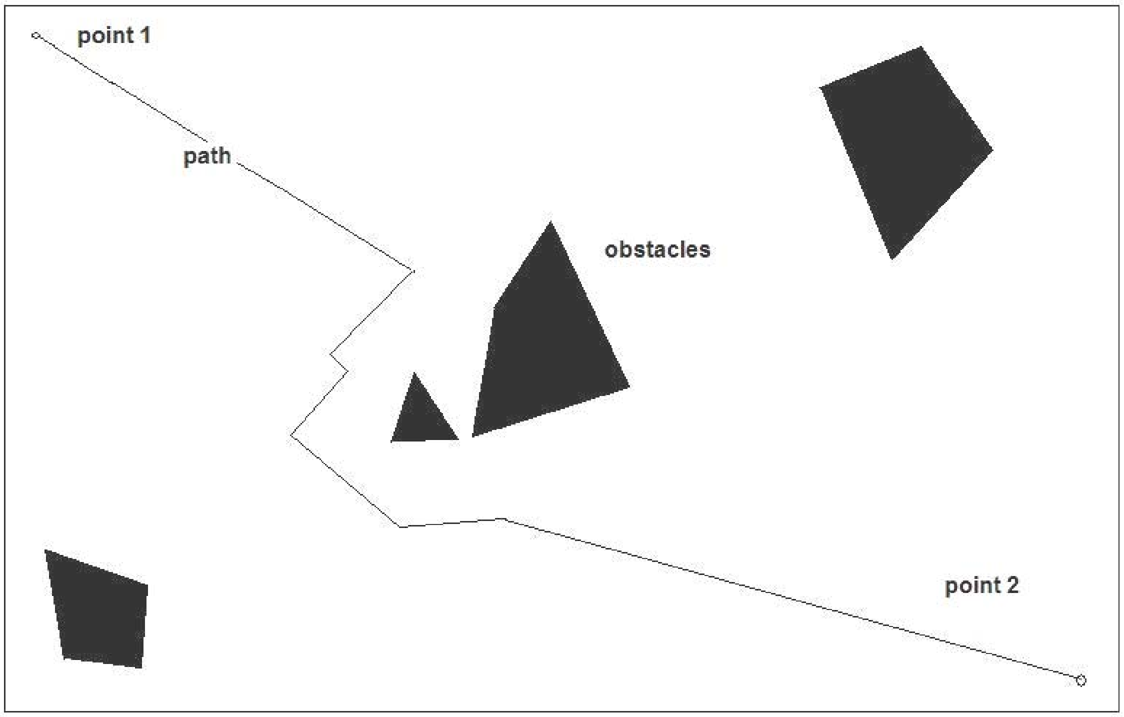


The base of the robot was constructed from a wooden ply. Two dc servo motors of 60 rpm, 12V were attached on the base using clamps. Tyres were then mounted on these dc motors .Two castor balls were also used. Each castor ball works like a ball bearing. The arrangement of the tyres and castor balls on the base of the robot was different from the general arrangement .Motors with tyres were attached on the base at two diagonally opposite corners of the rectangular base. The benefit of this arrangement was that the turning radius was decreased, the size of the robot was also decreased and the center of rotation lies on the base of the robot. Two thermocol pieces were also glued on the base to support the top cover (which has a circle and a rectangle of orange colour for identification) .A circuit was made on the PCB board which was implemented on the robot base using screws. In order to increase the friction between the tyres of the base and surface of the arena we glued cycle tyre tube on the tyre. A wireless circuit also placed on the base for wireless communication between computer and robot.12V dc battery (which consisted of 4 Nokia 2600 batteries of 3.7 V specifications) was also placed on robot base. The arena on which robot navigates was made up of many white chart papers attached together. This arena was of rectangular shape .Camera was placed at a height on the central axis of the arena to capture top view of the arena. Image was transferred from camera to the computer using data cable.

CHAPTER 7

RESULTS & CONCLUSION

Arena 7.1



The diagram shown in the previous page is the arena we used in our project .The robot moves from point 1 and goes along the path shown in the figure avoiding obstacles to the point 2 which is the destination point. When the robot was moving along its path it had some problems to navigate. Most important was that there was no friction between surface and robot. So in order to overcome this problem we glued cycle tyre tube on the tyres to increase friction.

Applications:

1. This type of robot can be used in extinguishing the fire
2. It can also be used handling waste material in nuclear reactions
3. It can be used in transportation in industries.

REFERENCES

1. Dudgeon, D.E. and R.M. Mersereau, Multidimensional Digital Signal Processing. 1984, Englewood Cliffs, New Jersey: Prentice-Hall.

2. Castleman, K.R., Digital Image Processing. Second ed. 1996, Englewood Cliffs, New Jersey: Prentice-Hall.

3. Oppenheim, A.V., A.S. Willsky, and I.T. Young, Systems and Signals. 1983, Englewood Cliffs, New Jersey: Prentice-Hall.

4. Papoulis, A., Systems and Transforms with Applications in Optics. 1968, New York: McGraw-Hill.

5. Russ, J.C., The Image Processing Handbook. Second ed. 1995, Boca Raton, Florida: CRC Press.

6. Giardina, C.R. and E.R. Dougherty, Morphological Methods in Image and Signal Processing. 1988, Englewood Cliffs, New Jersey: Prentice–Hall. 321.

7. Gonzalez, R.C. and R.E. Woods, Digital Image Processing. 1992, Reading, Massachusetts: Addison-Wesley. 716.

8. Goodman, J.W., Introduction to Fourier Optics. McGraw-Hill Physical and Quantum Electronics Series. 1968, New York: McGraw-Hill. 287.

9. Heijmans, H.J.A.M., Morphological Image Operators. Advances in Electronics and Electron Physics. 1994, Boston: Academic Press.

10. Hunt, R.W.G., The Reproduction of Colour in Photography, Printing & Television,. Fourth ed. 1987, Tolworth, England: Fountain Press.

11. Freeman, H., Boundary encoding and processing, in Picture Processing and Psychopictorics, B.S. Lipkin and A. Rosenfeld, Editors. 1970, Academic Press: New York. p. 241-266.

12. Stockham, T.G., Image Processing in the Context of a Visual Model. Proc.

IEEE, 1972. 60: p. 828 - 842.
13. Murch, G.M., Visual and Auditory Perception. 1973, New York: Bobbs-Merrill Company, Inc. 403.
14. Frisby, J.P., Seeing: Illusion, Brain and Mind. 1980, Oxford, England: Oxford University Press. 160.
15. Blakemore, C. and F.W.C. Campbell, On the existence of neurons in the human visual system selectively sensitive to the orientation and size of retinal images. J. Physiology, 1969. 203: p. 237-260.
16. Born, M. and E. Wolf, Principles of Optics. Sixth ed. 1980, Oxford: Pergamon Press.
17. Young, I.T., Quantitative Microscopy. IEEE Engineering in Medicine and Biology, 1996. 15(1): p. 59-66.
18. Dorst, L. and A.W.M. Smeulders, Length estimators compared, in Pattern Recognition in Practice II, E.S. Gelsema and L.N. Kanal, Editors. 1986, Elsevier Science: Amsterdam. p. 73-80.
19. Young, I.T., Sampling density and quantitative microscopy. Analytical and Quantitative Cytology and Histology, 1988. 10(4): p. 269-275.
20. Kulpa, Z., Area and perimeter measurement of blobs in discrete binary pictures. Computer Vision, Graphics and Image Processing, 1977. 6: p. 434-454.
21. Vossepoel, A.M. and A.W.M. Smeulders, Vector code probabilities and metrication error in the representation of straight lines of finite length. Computer Graphics and Image Processing, 1982. 20: p. 347–364.
22. Photometrics Ltd., Signal Processing and Noise, in Series 200 CCD Cameras Manual. 1990: Tucson, Arizona.
23. Huang, T.S., G.J. Yang, and G.Y. Tang, A Fast Two-Dimensional Median Filtering Algorithm. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1979. ASSP-27: p. 13-18.
24. Groen, F.C.A., R.J. Ekkers, and R. De Vries, Image processing with personal computers. Signal Processing, 1988. 15: p. 279-291.
25. Verbeek, P.W., H.A. Vrooman, and L.J. Van Vliet, Low-Level Image Processing by Max-Min Filters. Signal Processing, 1988. 15: p. 249-258.

26. Young, I.T. and L.J. Van Vliet, Recursive Implementation of the Gaussian Filter. Signal Processing, 1995. 44(2): p. 139-151.
27. Kuwahara, M., et al., Processing of RI-angiocardiographic images, in Digital Processing of Biomedical Images, K. Preston and M. Onoe, Editors. 1976, Plenum Press: New York. p. 187-203.
28. Van Vliet, L.J., Grey-scale measurements in multi-dimensional digitized images, PhD Thesis: Delft University of Technology, 1993.
29. Serra, J., Image Analysis and Mathematical Morphology. 1982, London: Academic Press.
30. Vincent, L., Morphological transformations of binary images with arbitrary structuring elements. Signal Processing, 1991. 22(1): p. 3-23.
31. Van Vliet, L.J. and B.J.H. Verwer, A Contour Processing Method for Fast Binary Neighbourhood Operations. Pattern Recognition Letters, 1988. 7(1): p. 27-36.
32. Young, I.T., et al., A new implementation for the binary and Minkowski operators. Computer Graphics and Image Processing, 1981. 17(3): p. 189-210
33. Lantuéjoul, C., Skeletonization in Quantitative Metallography, in Issues of Digital Image Processing, R.M. Haralick and J.C. Simon, Editors. 1980, Sijthoff and Noordhoff: Groningen, The Netherlands.
34. Oppenheim, A.V., R.W. Schafer, and T.G. Stockham, Jr., Non-Linear Filtering of Multiplied and Convolved Signals. Proc. IEEE, 1968. 56(8): p. 1264-1291.
35. Ridler, T.W. and S. Calvard, Picture thresholding using an iterative selection method. IEEE Trans. on Systems, Man, and Cybernetics, 1978. SMC-8(8): p. 630-632.
36. Zack, G.W., W.E. Rogers, and S.A. Latt, Automatic Measurement of Sister Chromatid Exchange Frequency. 1977. 25(7): p. 741-753.
37. Chow, C.K. and T. Kaneko, Automatic boundary detection of the left ventricle from cineangiograms. Computers and Biomedical Research, 1972. 5: p. 388-410.
38. Canny, J., A Computational Approach to Edge Detection. IEEE Transactions

on Pattern Analysis and Machine Intelligence, 1986. PAMI-8(6): p. 679-698.
39. Marr, D. and E.C. Hildreth, Theory of edge detection. Proc. R. Soc. London Ser. B., 1980. 207: p. 187-217.
40. Verbeek, P.W. and L.J. Van Vliet, On the Location Error of Curved Edges in Low-Pass Filtered 2D and 3D Images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1994. 16(7): p. 726-733.

41. Lee, J.S.L., R.M. Haralick, and L.S. Shapiro. Morphologic Edge Detection. in 8th International Conference on Pattern Recognition. 1986. Paris: IEEE Computer Society.
42. Van Vliet, L.J., I.T. Young, and A.L.D. Beckers, A Non-linear Laplace Operator as Edge Detector in Noisy Images. Computer Vision, Graphics, and Image Processing, 1989. 45: p. 167-195.
43. Meyer, F. and S. Beucher, Morphological Segmentation. J. Visual Comm. Image Rep., 1990. 1(1): p. 21-46.
44. Meyer, F., Iterative Image Transformations for an Automatic Screening of Cervical Cancer. Journal of Histochemistry and Cytochemistry, 1979. 27: p. 128-135.