**REPUBLIC OF RWANDA**

**KIGALI INDEPENDENT UNIVERSITY (ULK)**

**P.O Box 2280**

Website://www.ulk.ac.rw

E-mail: ulk@rwandatel.com



**COMPUTER GRAPHICS AND VISION**

**YEAR 4 COMPUTER SCIENCE**

**By:**

**Mr RWEGO MUCYO Placide**

**Tel. 0788496945**

**RUHUMURIZA Juru**

**0789515979**

**1. Module Code:** \_CSC 402**......... Faculty:** SCIENCE AND TECHNOLOGY

**2. Module Title:** COMPUTER GRAPHICS & VISION

**3. Level: 5 Semester: 1 Credits: 10**

**4. First year of presentation: 2008 Administering Faculty:** SCIENCE AND TECHNOLOGY

**5. Pre-requisite or co-requisite modules, excluded combinations**

**6. Allocation of study and teaching hours**

|  |  |  |
| --- | --- | --- |
| Total student hours: 100 | StudentHours | LectureHours |
| Lectures | 40 | 60 |
| Seminars/workshops |  |  |
| Practical classes/laboratory | 15 | 15 |
| Structured exercises | 10 | 10 |
| Set reading etc. | 10 |  |
| Self-directed study | 10 |  |
| Assignments – preparation and writing | 5 | 5 |
| Examination – revision and attendance | 10 | 10 |
| Other: (Invigilation & Marking) |  |  |
| TOTAL | 100 | 100 |

## 6.1 Brief description of aims and content

This module gives the opportunity to have a meaningful introduction to computer graphics. It also offers a quick, visual, and step-by-step approach for learning fundamental concepts of multimedia. It explains what multimedia is, how it works, and raises broader issues about how the various applications relate to the latest technology and jobmarket.

**6.2 Learning Outcomes**

**i) *Knowledge and Understanding***

Having successfully completed the module, students should be able to demonstrate knowledge and understanding of:

1. The wonder of modern computer graphics and show how the graphics process works.
2. The mechanisms by which we get computer graphics on the screen.
3. Multimedia authoring tools and recent developments
   1. ***Cognitive/Intellectual skills/Application of Knowledge***

Having successfully completed the module, students should be able to:

1. Pursue the development of graphics techniques and their associated interactive tools right up to the current state of the art.
2. Show for the potential user of graphics how the tools of graphic usage are created and thus how they may be exploited to advantage.
3. Think visually about problems
4. Transform the geometry into a suitable image
   1. ***Communication/ICT/Numeracy/Analytic Techniques/Practical Skills***

Having successfully completed the module, students should be able to:

1. Use a modern graphics API to create a graphics application
2. Integrate graphics into computer applications effectively
3. Have a more targeted goal such as games or scientific visualization
4. Create effective communications
5. Develop multimedia for a Web design firm.
   1. ***General transferable skills***

Having successfully completed the module, students should be able to:

1. Look forward to plenty of opportunity because the need for Multimedia and Design professionals is growing as fast as the Internet itself.
2. Carry out studies in graphic design and artistic technique
3. Create a visual representation that is to develop a geometrical representation of the model

**9. Indicative Content**

Characteristics of display devices: raster, vector, pixel etc. Representation of primitive objects :lines, curves, surfaces. Representation of composite objects. Graphics hardware: Continual refresh and storage displays. Devices resolutions. Display processors. Character generators. Display techniques: Colour-display techniques. Display description. Screen co-ordinates, user co-ordinates. Graphical data structures. Display-code generation. The viewing algorithm. Transformations. Graphics software: three dimensional graphics. Workstation models: bit-mapped, raster operations, postscript. Graphics Standards. Colour and texture: Shading and coloring; Ray tracing; Rendering processes; Animation techniques.

Multimedia - An Overview, Multimedia Elements: Text and Graphics. Understanding Multimedia Elements - Sound, Image and graphics, Animation, Video . Multimedia Authoring Programs. Data compression. Multimedia Operating Systems. Multimedia Communication Systems. Multimedia Applications. User interfaces. Development and Design of Multimedia Titles. Management and Distribution of Multimedia Titles. Documents, Hypertext and MHEG (Multimedia and Hypermedia Information Coding Expert Group). Case Study: Incorporating Multimedia into a Web Site., Interactive multimedia. Streaming, sampling techniques, Image recognition and digitization.

* + 1. **Learning and Teaching Strategy**

The lecturer will give clarifications so that students be able to go for research of more details. He has to give more assignment, group work and homework to students. He must try always to give case studies to students to let them digest well the subject and to know how to apply it to real life.

* + 1. **Assessment Strategy**

Formative and summative assessments are organized.

In-course assessment composed of written test, assignment or homework and handled practical assignment must be organized. Students have to receive comments on their works and results where it is needed.

**12. Indicative Resources**

**Core Text**

* Ralf Steinmetz, Klara Nahrstedt , [1995]*,**Multimedia Computing, Communications & Applications*, , Prentice Hall,
* Steve Cunningham , [2007], Computer Graphics, Programming, Problem Solving, and Visual Communication, , Printice Hall,

**Background Texts**

* Alan Watt and Steve Maddock, [2008],*3D Computer Graphics with OpenGL and Direct3D,* Fourth Edition, Addison Wesley,
* Donald Hearn, M. Pauline baker , [1997],*Computer graphics* 2nd Edition,C version, , Prentice Hall,

Computer Graphics and Vision

CSC402: COMPUTER GRAPHICS AND VISION 70 Hrs

Objective: After completing the module, all the students will have understood the concepts

of Computer Graphics and Computer Vision as well as the applications in imaging, digital visual

art and video game developments.

INDICATIVE CONTENTS

PART 1: COMPUTER GRAPHICS: 40Hrs

Line generation: points, lines, planes, pixels and frame buffers, vectors and character generation

Graphic primitives: Display Devices, Primitive Devices, Display file structure, Display control

text

Polygon: Polygon representation, entering polygon, filling polygons.

Segments: Segments table, creating, deleting and renaming segments, visibility, image

transformations.

Transformations: Matrices transformation, transformation routines, display procedures.

Windowing and clipping: Viewing transformation and clipping, generalize clipping, multiple

windowing

Three dimension: 3D geometry primitives, transformations, projection clipping.

Interaction: Hardware input devices handling algorithms, hidden line methods.

Rendering and Illumination: Introduction to curve generation, Bezier, Hermite and B-Spline

algorithms and their comparison

PART 2: COMPUTER VISION: 30Hrs

Cameras: Pinhole cameras, Cameras with Lenses, Human eye, Sensing, Radiometry, measuring

light: Light in space, Light at surfaces, Important special cases, Notes.

Source Shadows and shading: Qualitative radiometry, Sources and their effects, local shading.

Models, application: Photometric stereo, inter-reflections: Global shading models.

1

Computer Graphics and Vision

Color: The physics of color, Human color perception, Representing color, A model for image

color, Surface color from image color.

Linear filters: Linear filters and convolution, Shift Invariant linear systems, Shift invariant

linear system, Spatial frequency and Fourier Transforms, Sampling and aliasing, Filters and

templates.

Edge detection: Noise, estimating derivatives, Detecting edges.

Segmentation by clustering: What is segmentation? Human Vision: grouping and Gestalt.

Applications: Short boundary detection and background subtraction, Image segmentation by

clustering pixels, Segmentation by Graph-Theoretic Clustering.

Segmentation by fitting a model: Hough transform, Fitting lines, Fitting curves, Fitting as

probabilistic Inference Problem.

Tracking with Linear Dynamic Models: Tracking as an abstract Inference Problem, Linear

Dynamic Models, Kalman Filtering, Data Association.

TABLE OF CONTENTS

PART 1: COMPUTER GRAPHICS ..................................................................................................................... 4

1.

Introduction .......................................................................................................................................... 4

1.1 Overview ............................................................................................................................................. 4

1.2

Definition of some terms .............................................................................................................. 5

1.3 Applications of Computer Graphics .................................................................................................. 10

1.4 Graphics applications software......................................................................................................... 16

1.5 Character generation ........................................................................................................................ 17

1.6 Types of graphics .............................................................................................................................. 18

2. Video Display Devices ............................................................................................................................. 22

2.1 Refresh Cathode-Ray Tubes .............................................................................................................. 22

2.2 Raster-Scan Displays ......................................................................................................................... 24

2.3 Raster graphic Devices ...................................................................................................................... 26

2

Computer Graphics and Vision

2.4 Display FILE Structures ...................................................................................................................... 27

2.5 Random-Scan Displays ...................................................................................................................... 29

2.6 Raster-scan systems .......................................................................................................................... 30

2.7 Random-scan systems....................................................................................................................... 34

3. Graphics Monitors and Workstations ..................................................................................................... 34

3.1 Input devices ..................................................................................................................................... 35

3.2 Output devices .................................................................................................................................. 38

3.3 Image Printing Devices ...................................................................................................................... 39

3.4 Coordinate Representations ............................................................................................................. 40

3.5 Graphics Functions ............................................................................................................................ 40

4.

Graphic primitives ............................................................................................................................... 41

4.1 Points ................................................................................................................................................ 41

4.2 Lines .................................................................................................................................................. 42

4.3 Line generation algorithms ............................................................................................................... 44

4.4 Circle-generation algorithms ............................................................................................................ 64

4.5 Polygons ............................................................................................................................................ 74

5 Transformations ....................................................................................................................................... 80

5.1 2D Transformations .......................................................................................................................... 81

5.2 Combining Transformation ............................................................................................................... 82

5.3 Matrix representations and homogeneous coordinates .................................................................. 82

5.4 3D Transformations .......................................................................................................................... 84

5.5 Window-to-Viewport Coordinate Transformation ........................................................................... 88

5.6 Clipping operations ........................................................................................................................... 90

Part 2: COMPUTER VISION .......................................................................................................................... 97

6

Principles of Computer Vision ............................................................................................................. 97

6.1 Overview: .......................................................................................................................................... 97

6.2 Application of Computer vision: ..................................................................................................... 102

6.3 Camera ............................................................................................................................................ 104

6.4 CCD Cameras ................................................................................................................................... 116

6.5 Source, Shadow and Shading .......................................................................................................... 118

6.6 Color ................................................................................................................................................ 121

3

Computer Graphics and Vision

6.7 Linear filter ...................................................................................................................................... 124

6.8 Linear Shift-invariant Systems......................................................................................................... 125

6.9 Edge Detection ................................................................................................................................ 127

6.10. Shape Detection and Pattern Recognition................................................................................... 129

6.11 Segmentation ................................................................................................................................ 131

References ................................................................................................................................................ 135

Appendix: Table of Figures........................................................................................................................ 135

PART 1: COMPUTER GRAPHICS

1. Introduction

1.1 Overview

Computer Graphics is a branch of computer science that deals not only with producing

pictures or images using a computer, but also dealing with the theory and techniques of

computer image synthesis. The term computer graphics includes almost everything on

computers that is not text or sound. Computers have become a powerful tool for the rapid

and economical production of pictures. Advances in computer technology have made

interactive computer graphics a practical tool.

We find computer graphics to to display Information in design, simulation, computer Art,

Entertainment as diverse areas of science, engineering, medicine, business, industry,

government, education and training.

Computers produce images by analyzing a collection of dots, or pixels (picture elements).

Computer graphics facilitates the production of images that range in complexity from

simple line drawings to three-dimensional reconstructions of data obtained from

computerized axial tomography (CAT) scans in medical applications.

In the production of a computer-generated image, the designer has to specify the objects

in the image and their shapes, positions, orientations, and surface colors or textures.

Further, the viewer's position and direction of view (camera orientation) must be

4

Computer Graphics and Vision

specified. The software should calculate the parts of all objects that can be seen by the

viewer (camera). Only the visible portions of the objects should be displayed (captured

on the film). (This requirement is referred to as the hidden-surface problem.) The

rendering software is then applied to compute the amount and color of light reaching the

viewer eye (film) at any point in the image, and then to display that point. Some modern

graphics work stations have special hardware to implement projections, hidden-surface

elimination, and direct illumination.

1.2 Definition of some terms

1.2.1 Computer graphics

Computer graphics is a branch of computer science dealing with drawings or simply

graphics

created

using computers and,

more

generally,

the

presentation and manipulation of image data by a computer with help from specialized

software and hardware.

Computer graphics refers to any computer device or program that makes a computer

capable of displaying and manipulating pictures.

Computer graphics generally means creation, storage and manipulation of models and

images.

Computer graphics is just creating or manipulating images with computer.

It covers 4 areas:

1. Imaging = an image processing or representing 2Dimensional images

2. Modeling = representing 3Dimensional objects

3. Rendering = constructing 2Dimensional images from 3Dimensional models

4. Animation = simulating changes over time

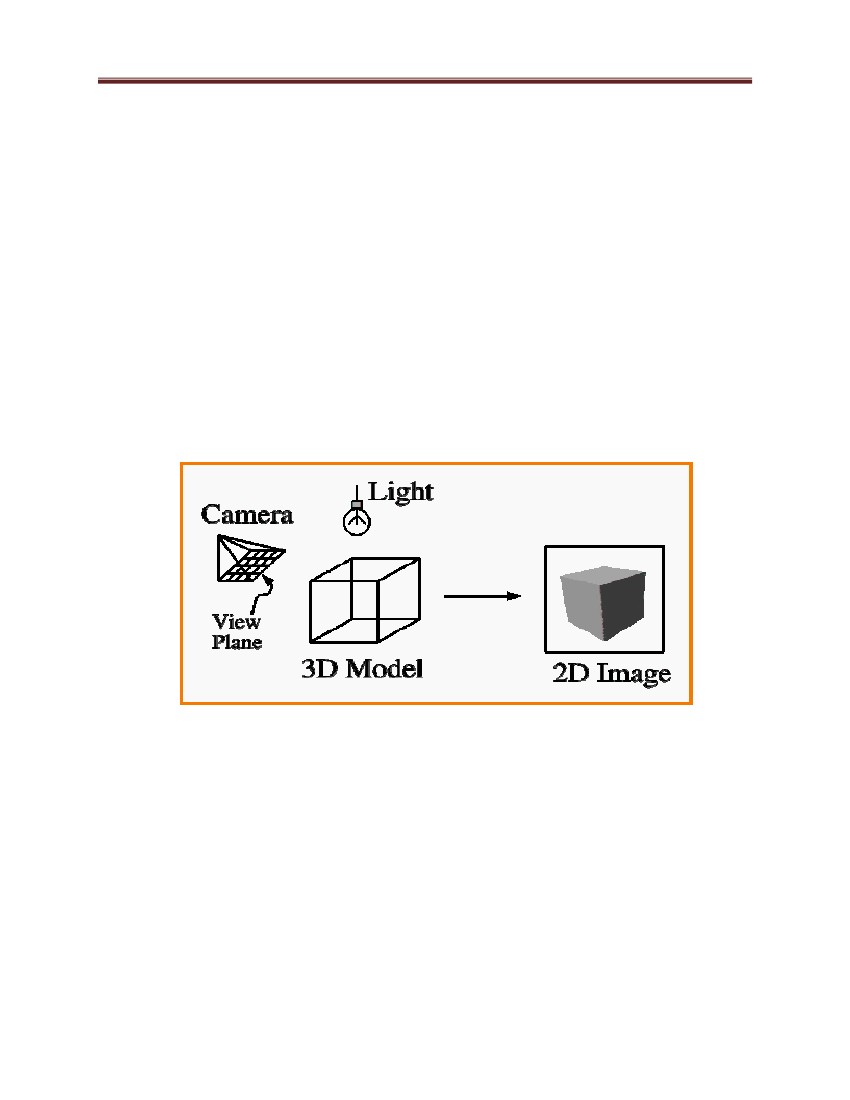
Imaging:

the same as Image processing. Computer imaging deals with storing,

manipulating, and transforming images with a computer. Images need first to have been

input by some means, such as a scanner, a video camera, a digital camera, etc.

5

Computer Graphics and Vision

Some computer graphics operations involve manipulating 2D images (bitmaps)

More generally, computer imaging deals with all aspects of processing images with a

computer, and encompasses image synthesis and analysis.

Common example include digital photo processing and digital ‗painting‘ programs

(Adobe Photoshop…)

Image synthesis or image generation refers more to the construction of images from

scratch, rather than processing of existing images.

Rendering: this is known as Synthesis of a 2D image from a 3D scene description that is

creating 2D image from 3D model creating images from models, ie applying physically

based procedures to generate images from scenes (using lighting and shading).

Figure 1: Rendering process

Modeling refers to the techniques involved with creating, scanning, editing, and

manipulating 3D geometric data. Modeling is often done by a human user with an

interactive editing program.

Animation is just a sequence of individual images. Basically, the subject of computer animation

focuses on how things change over time. Usually, this refers to motion, but can also refer to other

properties changing over time. Physical simulation is a very powerful tool in computer animation

and can be used to generate believable animations of rigid objects, deformable objects, gasses,

6

Computer Graphics and Vision

liquids, fracture, particle effects, and even explosions and fire. Computer animation also includes

a large number of techniques specifically developed to manipulate virtual characters.

1.2.2 Computer vision

Computer vision is a field that includes methods for acquiring, processing, analyzing, and

understanding images and, in general, high-dimensional data from the real world in order to

produce numerical or symbolic information, e.g., in the forms of decisions. A theme in the

development of this field has been to duplicate the abilities of human vision by electronically

perceiving and understanding an image. Computer vision is sometimes considered as a separate

discipline from computer graphics, although they share many things in common. A central goal

in computer vision is to take a set of 2D images (usually from a video or set of photos) and infer

from that a 3D description of what is being viewed. This is a very different process than

rendering, and is more of a form of artificial intelligence.

1.2.3 Pixels

In every image you have pixels. These are the tiny little dots of color you see on your

screen, and the smallest possible size any image can get. When an image is stored, the

image file contains information on every single pixel in that image.

Pixel (picture element) is one of the individual dots that make up a graphical image, each

of which combines red, green, and blue (RGB) phosphors to create a specific color. A

VGA screen in high-resolution mode consists of 640 X 480 or 307,200 pixels. Pixel is the

smallest resolved unit of a video image that has specific luminescence and color. Its

proportions are determined by the number of lines making up the scanning raster (the

pattern of dots that form the image) and the resolution along each line. In the most

common form of computer graphics, the thousands of tiny pixels that make up an

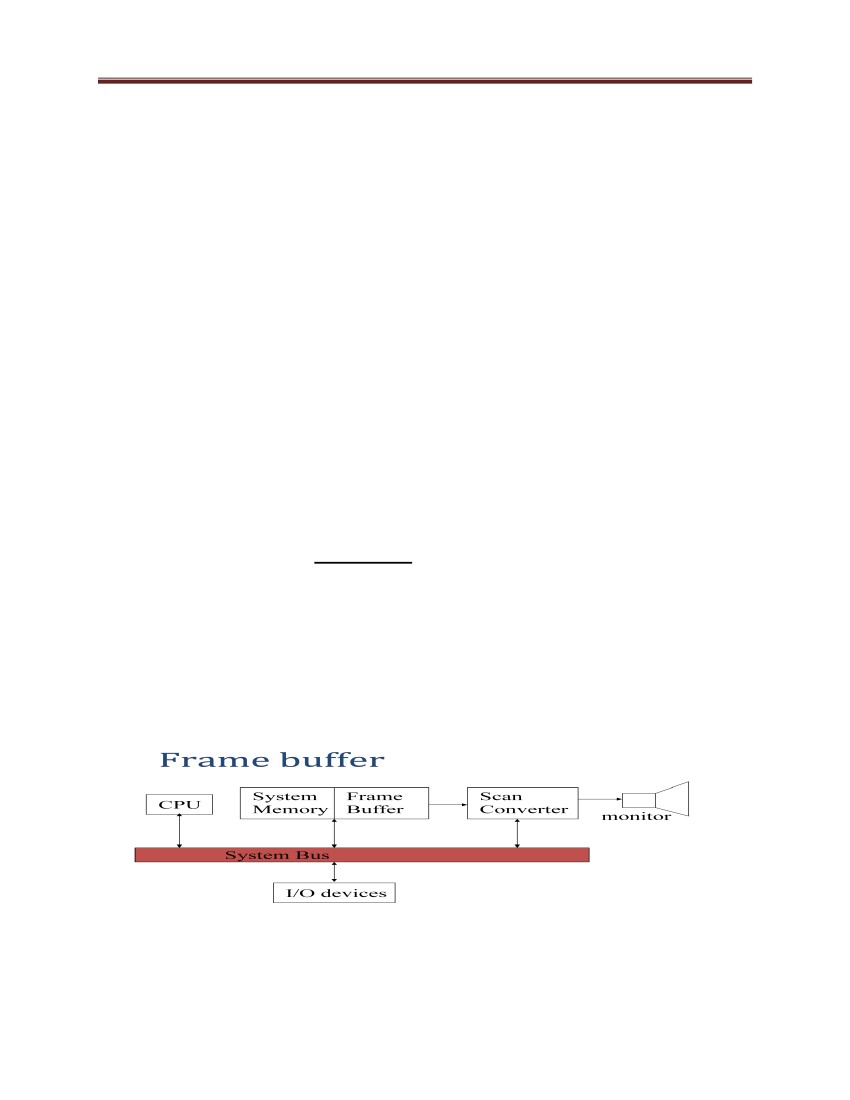
individual image are projected onto a display screen as illuminated dots that from a

distance appear as a continuous image. An electron beam creates the grid of pixels by

tracing each horizontal line from left to right, one pixel at a time, from the top line to the

bottom line.

7

Computer Graphics and Vision

1.2.4 Resolution

Images also have a set number of pixels per size of the image, known as resolution. You

might see terms such as dpi (dots per square inch), meaning the number of pixels you will

see in a square inch of the image. A higher resolution means there are more pixels in a set

area, resulting in a higher quality image. The disadvantage of higher resolution is that it

requires more processing power to analyze an image. When programming computer

vision into a robot, use low resolution.

In a general term it is referred to image resolution, monitor resolution and output device

resolution. To be precise, it is a unit of measurement, used to determine the size of an

image, the way an image displayed on the monitor and the device on which an image is

output.

1.2.5 The frame buffer

The frame buffer refers to the memory dedicated to storing the image. - It would

generally be a 2D array of pixels, where each pixel stores a color (Note: pixel = picture

element)

A frame buffer can just be a block of main memory, but many graphics systems have

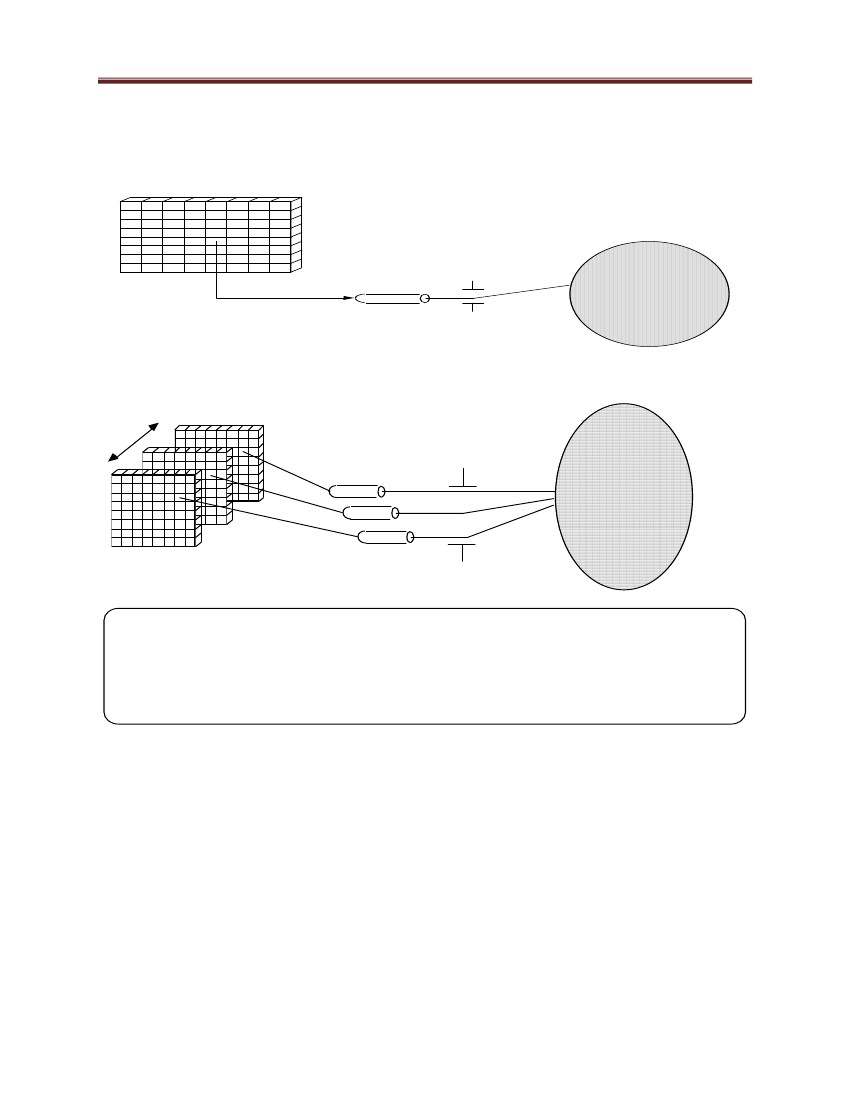
dedicated frame buffer memory with a direct connection to video scan-out hardware and

other special features.

Figure 2: Graphics program executes in CPU, filling the frame buffer with values to

display

8

Computer Graphics and Vision

The RGB Color Model

There are separate frame buffers for each of our basic colors (Red, Green, and Blue). It

requires a lot of memory for the frame buffer.

1 bit

2 le v e ls

Ele ctron

Gun

Figure 3: 1-Bit Memory, Monochrome Display (Bitmap Display)

3

re d

gre e n

blue

COLOR: black re d gre e n blue ye llow c yan mage nta white

R

G

B

0

0

0

1

0

0

0

1

0

0

0

1

1

1

0

0

1

1

1

0

1

1

1

1

Figure 4: 3 Bit color Display

The color is typically stored as a 24 bit RGB value. This offers 8 bits (256 levels) for red, green,

and blue, for a total of 16,777,216 different colors

24 bitplanes, 8 bits per color gun.

224 = 16,777,216

9

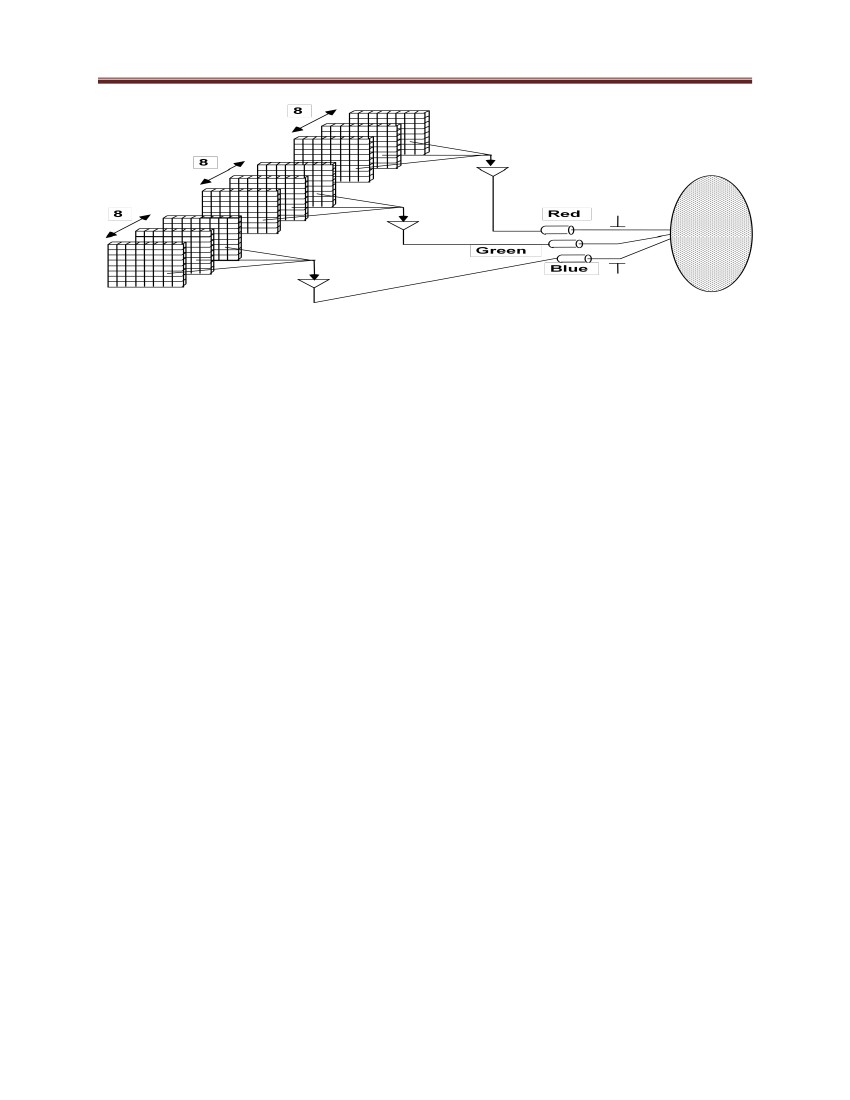
Computer Graphics and Vision

Figure 5: True color display

1.2.6 Raster Graphics

Modern graphics displays are raster based. This just means that they display a grid of pixels,

where each pixel color can be set independently. Individual pixels are usually formed from

smaller red, green, and blue sub pixels. If you look very closely at a TV screen or computer

monitor, you will notice the pattern of sub pixels. Older style vector displays didn‘t display a

grid of pixels, but instead drew lines directly with an electron beam. Raster graphics are also

sometimes called bitmapped Graphics

1.2.7 Primitives

A basic building block for graphic images, such as a dot, line, square or curve, Complex scenes are

usually built up from simpler objects

Objects are built from individual primitives. The most common and general purpose 3D primitive is

the triangle. Points and lines are also useful primitives.

1.3 Applications of Computer Graphics

1.3.1 Computer Aided Design

A major use of computer graphics is in design processes, particularly for engineering and

architectural systems, but almost all products are now computer designed.

10

Computer Graphics and Vision

Figure 6: Los Angeles City by use of CAD process

Generally referred to as CAD, computer-aided design methods are now routinely used in

the design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles,

and many, many other products.

1.3.2 Presentation graphics

Another major application is presentation graphics, used to produce illustrations for

reports or to generate 35-mm slides or transparencies for use with projectors. Presentation

graphics is commonly used to summarize financial, statistical, mathematical, scientific,

and economic data for research reports, managerial reports, consumer information

bulletins, and other types of reports. Workstation devices and service bureaus exist for

converting screen displays into 35-mm slides or overhead transparencies for use in

presentations. Typical examples of presentation graphics are bar charts, line graphs,

surface graphs, pie charts, and other displays showing relationships between multiple

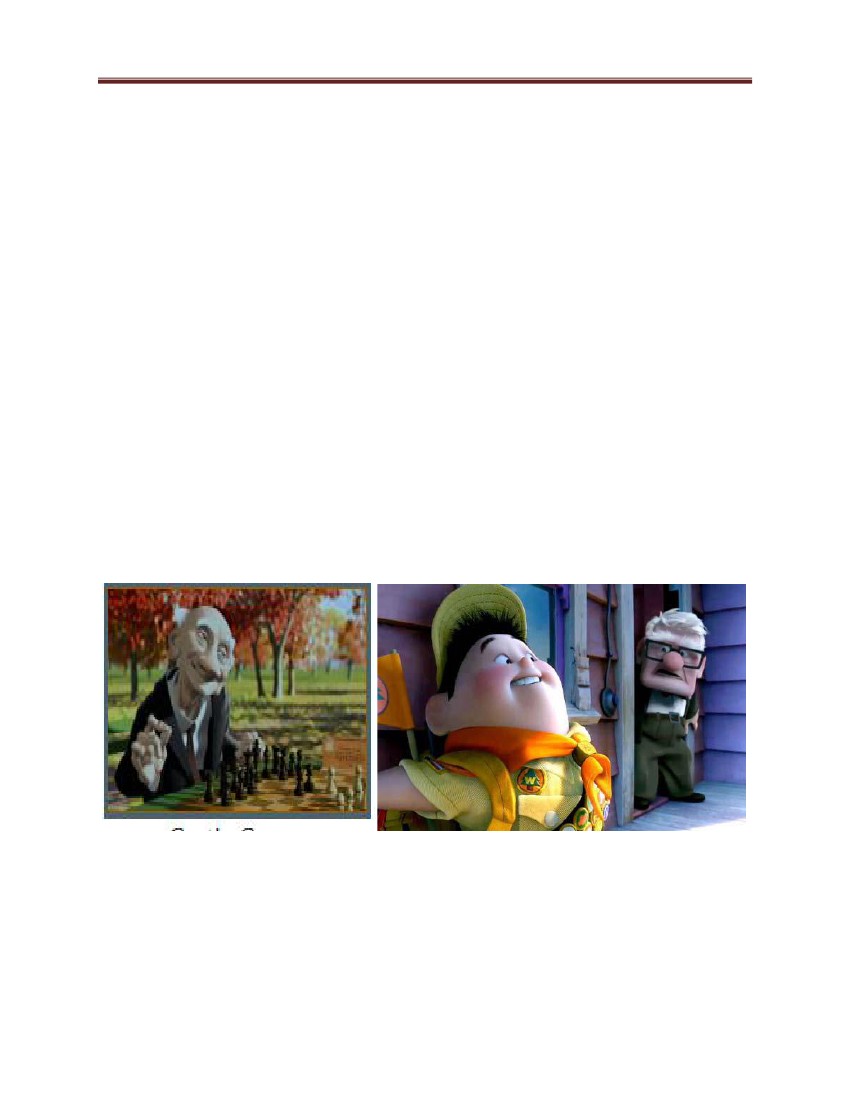
parameters.

1.3.3 Computer art

Computer graphics methods are widely used in both fine art and commercial art

applications. Artists use a variety of computer methods, including special-purpose e

11

Computer Graphics and Vision

hardware, artist's paintbrush program (such as Lumena), other paint packages (such as

Pixel paint and Super paint), specially developed software, symbolic mathematic

packages (such as Mathematics), CAD packages, desktop publishing

software, and animation packages that provide facilities for designing object shapes and

specifying object motions.

1.3.4 Entertainment

Computer graphics methods become now commonly used in making motion pictures,

music videos, and television shows. Sometimes the graphics scenes are displayed by

themselves, and sometimes graphics objects are combined with the actors and live

scenes. Many TV series regularly employ computer graphics methods. Music videos use

graphic in several ways. Graphics objects can be combined with the live action, or

graphics and image processing techniques can be used to produce a transformation of one

person or object into another (morphing). An example of morphing is shown in the

sequence of scenes.

Figure 7: Example of Entertainment graphics

12

Computer Graphics and Vision

1.3.5 Education and Training

Computer-generated models of physical, financial, and economic systems are often used

as educational aids. Models of physical systems, physiological systems, population

trends, or equipment can help trainees to understand the operation of the system.

For some training applications, special systems are designed. Examples of such

specialized systems are the simulators for practice sessions or training of ship captains,

aircraft pilots, heavy-equipment operators, and air traffic control personnel. Some

simulators have no video screens; for example, a flight simulator with only a control

panel for instrument flying. But most simulators provide graphics screens for visual

operation.

Human Skeleton

Flight simulation

Figure 8: Example of Educational Graphics

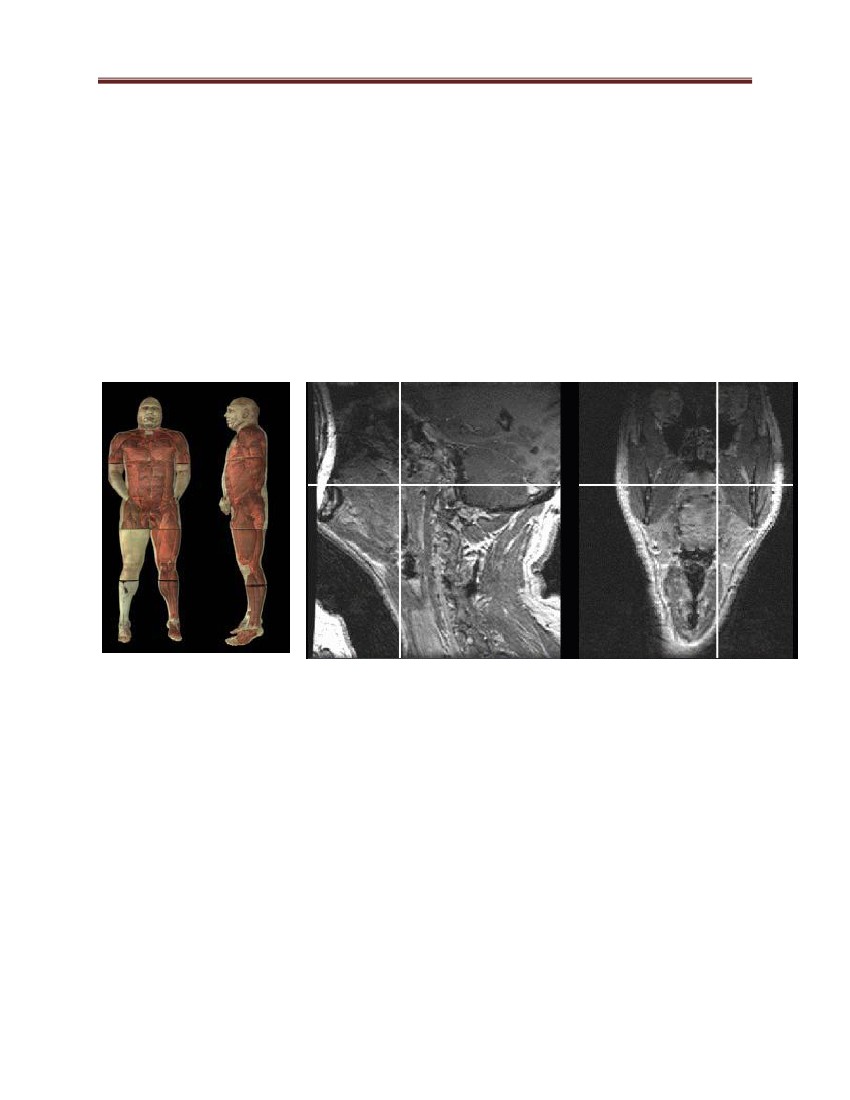
1.3.6 Visualization

Scientists, engineers, medical personnel, business analysts, and others often need to

analyze large amounts of information or to study the behavior of certain processes.

Numerical simulations carried out on supercomputers frequently produce data files

13

Computer Graphics and Vision

containing thousands and even millions of data values. Similarly, satellite cameras and

other sources are amassing large data files faster than they can be interpreted. Scanning

these large sets of number to determine trends and relationships is a tedious and

ineffective process. But if the data are converted to a visual form, the trends and patterns

are often immediately apparent.

Mathematicians, physical scientists, and others use visual techniques to analyze

mathematical functions and processes or simply to produce interesting graphical

representation.

Visible Human

Image-Guided Surgery Project

Figure 9: Examples of Visualization Graphics

1.3.7 Image processing

Although methods used in computer graphics and Image processing overlap, the two

areas are concerned with fundamentally different operations. In computer graphics, a

computer is used to create a picture. Image processing, on the other hand, applies

techniques to modify or interpret existing pictures, such as photographs and TV scans.

Two principal applications of image processing are (1) improving picture quality and (2)

machine perception of visual information, as used in robotics.

14

Computer Graphics and Vision

To apply image processing methods, we first digitize a photograph or other picture into

an image file. Then digital methods can be applied to rearrange picture parts, to enhance

color separations, or to improve the quality of shading. These techniques are used

extensively in commercial art applications that involve the retouching and rearranging of

sections of photographs and other artwork. Similar methods are used to analyze satellite

photos of the earth and photos of galaxies.

Medical applications also make extensive use of image processing techniques for picture

enhancements, in tomography and in simulations of operations. Tomography is a

technique of X-ray photography that allows cross-sectional views of physiological

systems to be displayed. Both computed X-rav tomography (CT) and position emission

tomography (PET) use projection methods to reconstruct cross sections from digital data.

These techniques are also used to monitor internal functions and show cross sections

during surgery. Other medical imaging techniques include ultrasonic and

nuclear

medicine scanner, With ultrasonic, high frequency sound waves, instead of X-rays, are

used to generate digital data.

Nuclear medicine scanners collect digital data from radiation emitted from ingested

radionuclide and plot color-coded images.

Image processing and computer graphics are typically combined in many applications:

medicine, for example uses these techniques to model and study physical functions, to

design artificial limbs and to plan and practice surgery. The last application is generally

referred to as Computer aided surgery.

Two-dimensional cross sections of the body are obtained using imaging techniques. Then

the slices are viewed and manipulated using graphics methods to simulate actual surgical

procedures and to try out different surgical cuts.

1.3.8 Graphical User Interfaces (GUI)

It is common now for software packages to provide a graphical interface. A major

component of a graphical interface is a window manager that allows a user to display

multiple-window areas. Each window can contain a different process that can contain

15

Computer Graphics and Vision

graphical or non graphical displays. To make a particular window active, we simply click

in that window using an interactive pointing device.

Interfaces also display menus and icons for fast selection of processing options or

parameter values. An icon is a graphical symbol that is designed to look like the

processing option it represents. The advantages of icons are that they take up less screen

space than corresponding textual descriptions and they can be understood more quickly if

well designed.

1.4 Graphics applications software

There are two general classifications for graphics software: general programming packages and

special-purpose application packages. A general graphics programming package provides an

extensive set of graphics functions that can be used in a high-level programming language, such

as C or FORTRAN. An example of a general graphics programming package is the GL

(Graphics Library) system on Silicon Graphics equipment. Basic functions in a general package

include those for generating picture components (straight lines, polygons, circles, and

other figures), setting color and intensity values, selecting views, and applying transformations.

By contrast, application graphics packages are designed for nonprogrammers, so that users can

generate displays without worrying about how graphics operations work. The interface to the

graphics routines in such packages allows users to communicate with the programs in their own

terms. Examples of such applications packages are the artist's painting programs and various

business, medical, and CAD systems.

Some examples:

- paint programs : Allow you to create rough freehand drawings. The images are stored

as bit maps and can easily be edited.

- illustration/design programs: Supports more advanced features than paint programs,

particularly for drawing curved lines. The images are usually stored in vector-based

formats. Illustration/design programs are often called draw programs.

16

Computer Graphics and Vision

- presentation graphics software : Lets you create bar charts, pie charts, graphics, and

other types of images for slide shows and reports. The charts can be based on data

imported from spreadsheet applications.

- animation software: Enables you to chain and sequence a series of images to simulate

movement. Each image is like a frame in a movie.

- CAD software: Enables architects and engineers to draft designs.

-desktop publishing : Provides a full set of word-processing features as well as fine

control over placement of text and graphics, so that you can create newsletters,

advertisements, books, and other types of documents.

In general, applications that support graphics require a powerful CPU and a large amount

of memory. Many graphics applications—for example, computer animation systems—

require more computing power than is available on personal computers and will run only

on powerful workstations or specially designed graphics computers. This is true of all

three-dimensional computer graphics applications.

In addition to the CPU and memory, graphics software requires a graphics monitor and

support for one of the many graphics standards. Most PC programs, for instance, require

VGA graphics. If your computer does not have built-in support for a specific graphics

system, you can insert a video adapter card.

The quality of most graphics devices is determined by their resolution—how many points

per square inch they can represent—and their color capabilities.

1.5 Character generation

Letters, numbers, and other characters can be displayed in a variety of sizes and styles. The

overall design style for a set (or family) of characters is called a typeface. Today, there are

hundreds of typefaces available for computer applications.

Examples of a few common

typefaces are Courier, Helvetica, New York, Palatino, and Zapf Chancery. Originally, the term

font referred to a set of cast metal character forms in a particular size and format, such as 10-

point Courier Italic or 12-point Palatino Bold.

17

Computer Graphics and Vision

Two different representations are used for storing computer fonts. A simple method for

representing the character shapes in a particular typeface is to use rectangular grid patterns. The

set of characters are then referred to as a bitmap font (or bitmapped font). Another, more

flexible, scheme is to describe character shapes using straight-line and curve sections,

When the pattern is copied to an area of the frame buffer, the 1 bits designate which pixel

positions are to be displayed on the monitor.

Bitmap fonts are the simplest to define and display: The character grid only needs to be mapped

to a frame-buffer position. In general, however, bitmap fonts require more space, because each

variation (size and format) must be stored in a font cache. It is possible to generate different sizes

and other variations, such as bold and italic, from one set, but this usually does not produce good

results. In contrast to bitmap fonts, outline fonts require less storage since each variation does

not require a distinct font cache. We can produce bold face, italic, or different sizes by

manipulating the curve definitions for the character outlines. But it does take more time to

process the outline fonts, because they must be scan converted into the frame buffer.

1.6 Types of graphics

1.6.1 Bitmap Images

Bitmap images (also known as raster images) are made up of pixels in a grid. Pixels are picture

elements; tiny dots of individual color that make up what you see on your screen. All these tiny

dots of color come together to form the images you see. Most computer monitors display

approximately 70 to 100 pixels per inch--the actual number depends on your monitor and screen

settings.

To illustrate this, let's take a look at a typical desktop icon such as the one shown in the image

here. The icons on your desktop are typically 32 by 32 pixels. In other words, there are 32 dots of

color going in each direction. When combined, these tiny dots form an image. The icon shown in

the upper right corner of this example is a typical desktop icon at screen resolution. As you can

see, when you enlarge the icon, you can clearly see each individual square dot of color. Note the

that white areas of the background are still individual pixels, even though they appear to be one

solid color. Figure 10 shows the bitmap image which is zoomed with the original size in the

upper right corner.

18

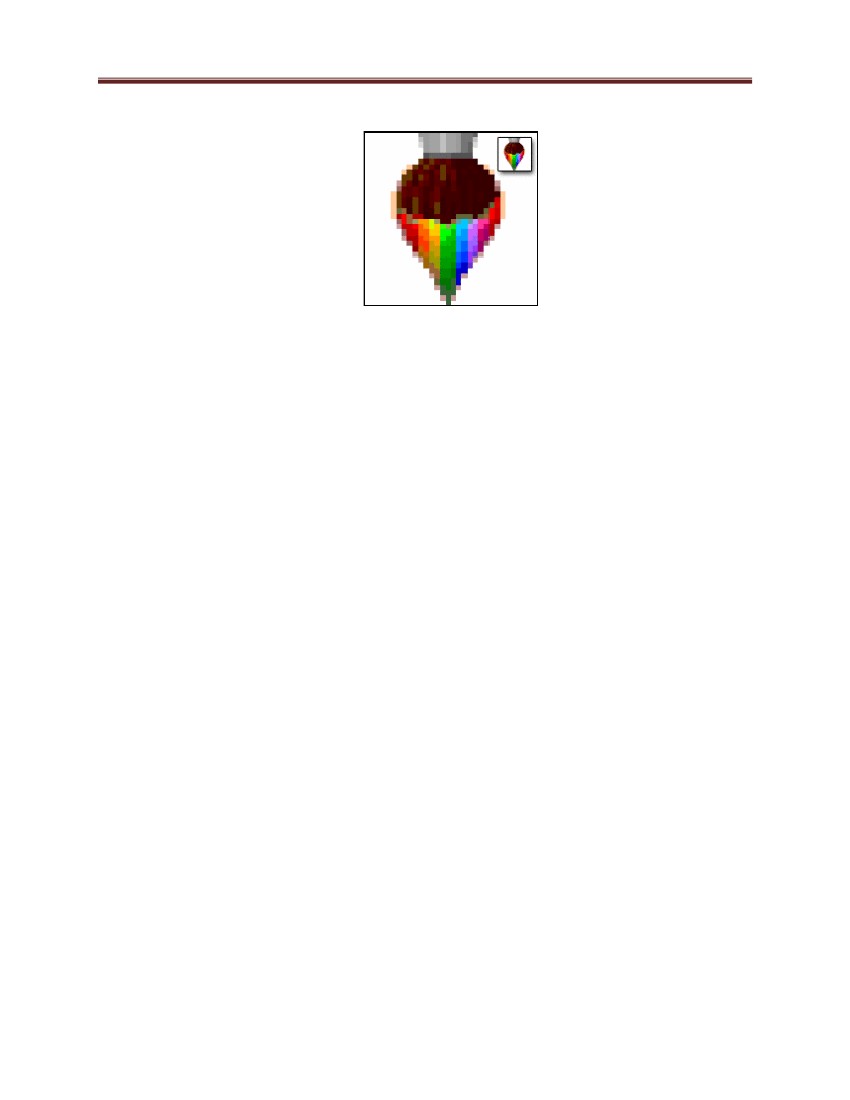
Computer Graphics and Vision

Figure 10: A Bitmap Image-

Bitmap images are resolution dependent. Resolution refers to the number of pixels in an image

and is usually stated as dpi (dots per inch) or ppi (pixels per inch). Bitmap images are displayed

on your computer screen at screen resolution: approximately 100 ppi. However, when printing

bitmaps, your printer needs much more image data than a monitor. In order to render a bitmap

image accurately, the typical desktop printer needs 150-300 ppi. If you've ever wondered why

your 300 dpi scanned image appears so much larger on your monitor, this is why.

Because bitmaps are resolution dependent, it's difficult to increase or decrease their size without

sacrificing a degree of image quality. When you reduce the size of a bitmap image through your

software's resample or resize command, you must throw away pixels. When you increase the size

of a bitmap image through your software's resample or resize command, the software has to

create new pixels. When creating pixels, the software must estimate the color values of the new

pixels based on the surrounding pixels. This process is called interpolation.

I want to make the distinction between this method of resizing versus zooming in and out, or

dragging the edges of your images in a page layout program to resize it. This type of resizing is

more accurately called scaling. Scaling an image does not affect the image permanently. In other

words, it does not change the number of pixels in the image. However, if you scale a bitmap

image to a larger size in your page layout software, you are going to see a definite jagged

appearance. Even if you don't see it on your screen, it will be very apparent in the printed image.

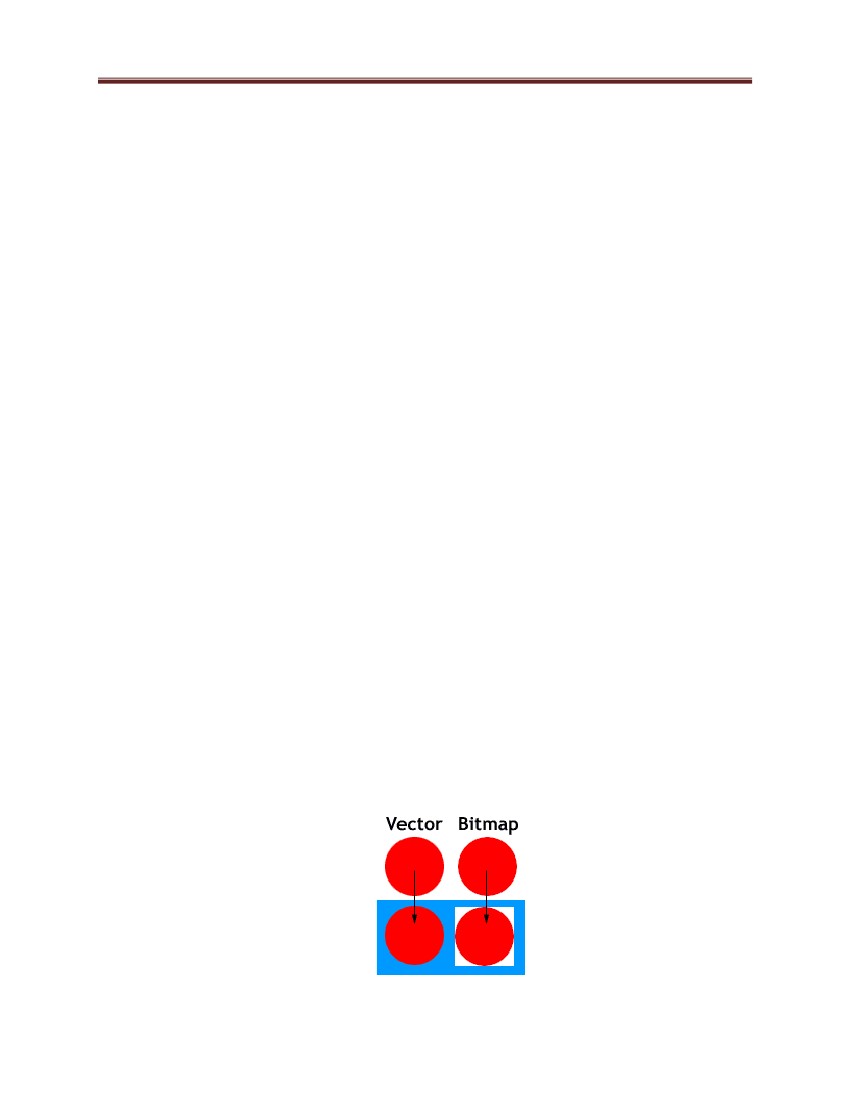
Scaling a bitmap image to a smaller size doesn't have any effect; in fact, when you do this you

are effectively increasing the ppi of the image so that it will print clearer.

Common bitmap formats include:

• BMP • GIF • JPEG, JPG • PNG • PICT (Macintosh) • PCX • TIFF • PSD (Adobe Photoshop)

19

Computer Graphics and Vision

Popular bitmap editing programs are: • Microsoft Paint • Adobe Photoshop • Corel Photo-

Paint • Corel Paint Shop Pro • The GIMP

All scanned images are bitmaps, and all images from digital cameras are bitmaps.

Converting between bitmap formats is generally as simple as opening the image to be converted

and using your software's Save As... command to save it in any other bitmap format supported by

your software.

Key Points about Bitmap Images: • pixels in a grid • resolution dependent • resizing reduces

quality • easily converted • restricted to rectangle • minimal support for transparency

1.6.2 Vector graphics

Vector images are made up of many individual, scalable objects. These objects are defined by

mathematical equations rather than pixels, so they always render at the highest quality. Objects

may consist of lines, curves, and shapes with editable attributes such as color, fill, and outline.

Changing the attributes of a vector object does not affect the object itself. You can freely change

any number of object attributes without destroying the basic object.

An object can be modified not only by changing its attributes, but also by shaping and

transforming it using nodes and control handles.

Because they're scalable, vector-based images are resolution independent. You can increase and

decrease the size of vector images to any degree and your lines will remain crisp and sharp, both

on screen and in print. Fonts are a type of vector object.

Another advantage of vector images is that they're not restricted to a rectangular shape like

bitmaps. Vector objects can be placed over other objects, and the object below will show

through. See the example images on this page. The vector circle and bitmap circle appear to be

exactly the same when seen on a white background. But when you place the bitmap circle over

another color, it has a rectangular box around it, from the white pixels in the image.

20

Computer Graphics and Vision

Figure 11: Vector and Bitmap Differences-vector images are not confined to a rectangular

shape.

Vector images have many advantages, but the primary disadvantage is that they're unsuitable for

producing photo-realistic imagery. Vector images are usually made up of solid areas of color or

gradients, but they cannot depict the continuous subtle tones of a photograph. That's why most of

the vector images you see tend to have a cartoon-like appearance

Vector images primarily originate from software. You can't scan an image and save it as a vector

file without using special conversion software. On the other hand, vector images can, quite

easily, be converted to bitmaps. This process is called rasterizing. When you convert a vector

image to a bitmap, you can specify the output resolution of the final bitmap for whatever size

you need. It's always important to save a copy of your original vector artwork in its native format

before converting it to a bitmap; once it has been converted to a bitmap, the image loses all the

wonderful qualities it had in its vector state. If you convert a vector to a bitmap at a size of 100

by 100 pixels and then decide you need the image to be larger, you'll need to go back to the

original vector file and export the image again. Also keep in mind that opening a vector image in

a bitmap editing program usually destroys the vector qualities of the image and converts it to

raster data.

The most common reason for wanting to convert a vector to a bitmap would be for use on the

Web. At this time, the most common and accepted format for vector images on the Web is

Shockwave Flash (SWF). Another standard for vector images on the Web is SVG, a graphics

programming language based on XML. Due to the nature of vector images, they are best

converted to GIF or PNG format for use on the Web.

Common vector formats include: • AI (Adobe Illustrator) • CDR (CorelDRAW) • CMX (Corel

Exchange) • CGM Computer Graphics Metafile

• DXF AutoCAD • WMF Windows Metafile

Popular vector drawing programs are: • Adobe Illustrator • CorelDRAW • Xara Xtreme •

Serif DrawPlus

21

Computer Graphics and Vision

2. Video Display Devices

Typically, the primary output device in a graphics system is a video monitor (Figure 12). The

operation of most video monitors is based on the standard cathode-ray

tube (CRT) design, but several other technologies exist and solid-state monitors

may eventually predominate

Figure 12 A computer graphics workstation

The size of a display is typically given as the distance between two opposite screen

corners. One problem with this method is that it does not distinguish between the aspect

ratios of monitors with identical diagonal sizes, in spite of the fact that a shape of a given

diagonal span's area decreases as it becomes less square. For example, a 4:3 21" monitor

has an area of ~211 square inches, while a 16:9 21" widescreen has an area of only ~188

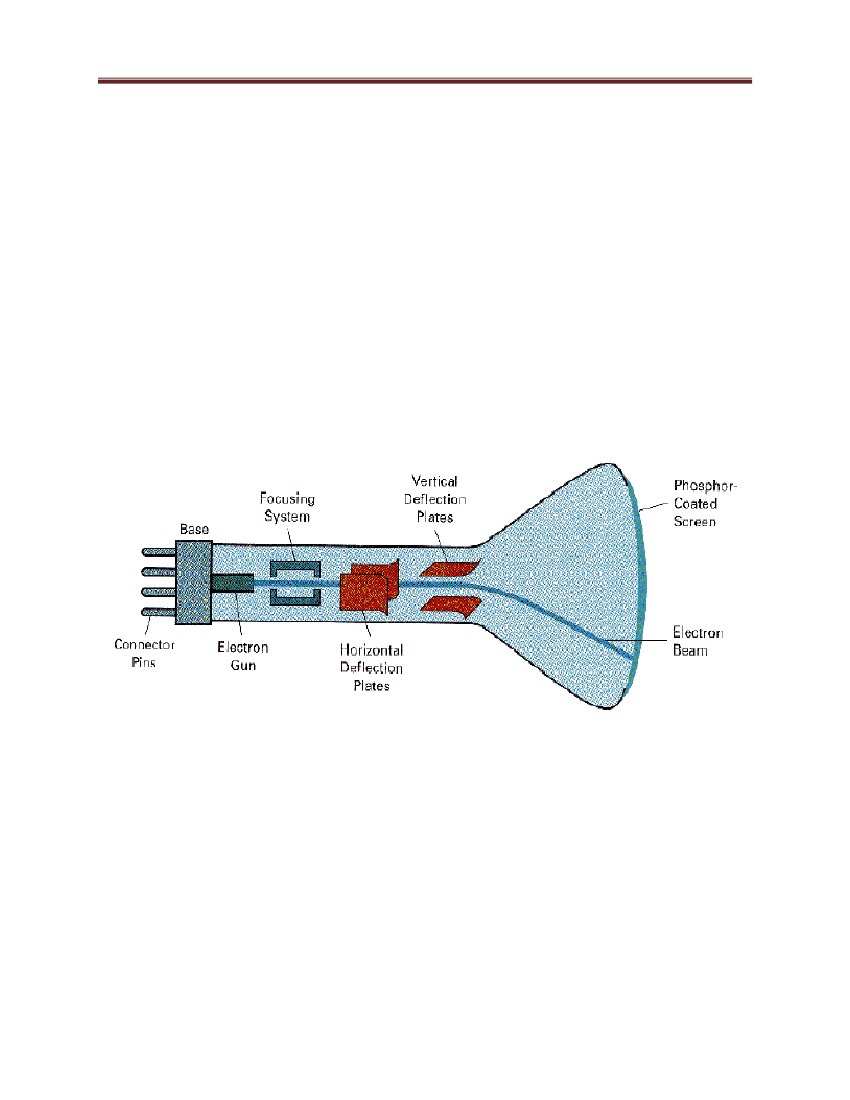
square inches

2.1 Refresh Cathode-Ray Tubes

Fig 7-2 illustrates the basic operation of a CRT. A beam of electrons (cathode rays) emitted by

an electron gun, passes through focusing and deflection systems that direct the beam toward

22

Computer Graphics and Vision

specified positions on the

phosphor

–coated screen. The phosphor then emits a small spot of light

at each position contacted by the electron beam. Because the light emitted by the phosphor fades

very rapidly, some method is needed for maintaining the screen picture. One way to keep the

phosphor glowing is to redraw the picture repeatedly by quickly directing the electron beam back

over the same points. This type of display is called a refresh CRT.

The primary components of an electron gun in a CRT are the heated metal cathode and a control

grid . Heat is supplied to the cathode by directing a current through a coil of wire, called the

filament, inside the cylindrical cathode structure. This causes electrons to be 'killed off" the hot

cathode surface. In the vacuum inside the CRT envelope, the free, negatively charged electrons

are then accelerated toward the phosphor coating by a high positive voltage. The accelerating

voltage can be generated with a positively charged metal coating

Figure 13: Basic operation of a CRT

On the side of the CRT envelope near the phosphor screen or an accelerating anode can be used.

Intensity of the electron beam is controlled by setting voltage levels on the control grid, which is

a metal cylinder that fits over the cathode. A high negative voltage applied to the control grid

will shut off the beam by repelling electrons and stopping them from passing through the small

hole at the end of the control grid structure. A smaller negative voltage on the control grid simply

decreases the number of electrons passing through. Since the amount of light emitted by

23

Computer Graphics and Vision

the phosphor coating depends on the number of electrons striking the screen, we control the

brightness of a display by varying the voltage on the control grid.

The maximum number of points that can be displayed without overlap on a CRT is referred to as

the resolution. A more precise definition of resolution is the number of points per centimeter

that can be plotted horizontally and vertically, although it is often simply stated as the total

number of points in each direction.

The resolution of a CRT is dependent on the type of phosphor, the intensity to be displayed, and

the focusing and deflection systems. Typical resolution on high-quality systems is 1280 by 1024,

with higher resolutions available on many systems. High resolution systems are often referred to

as high-definition systems. The physical size of a graphics monitor is given as the length of the

screen diagonal, with sizes varying from about 12 inches to 21 inches or more. A CRT monitor

can be attached to a variety of computer systems, so the number of screen points that can

actually be plotted depends on the capabilities of the system to which it is attached.

2.2 Raster-Scan Displays

The most common type of graphics monitor employing a CRT is the raster-scan display, based

on television technology. In a raster-scan system, the electron beam is swept across the screen,

one row at a time from top to bottom. As the electron beam moves across each row, the beam

intensity is turned on and off to create a pattern of illuminated spots. Picture definition is stored

in a memory area called the refresh buffer or frame buffer. This memory area holds the set of

intensity values for all the screen points. Stored intensity values are then retrieved from the

refresh buffer and "painted" on the screen one row (scan line) at a time (Figure 14). Each screen

point is referred to as a pixel or pel (shortened forms of picture element). The capability of a

raster-scan system to store intensity information for each screen point makes it well suited for the

realistic display of scenes containing subtle shading and color patterns. Home television sets and

printers are examples of other systems using raster-scan methods

Intensity range for pixel positions depends on the capability of the raster system. In a simple

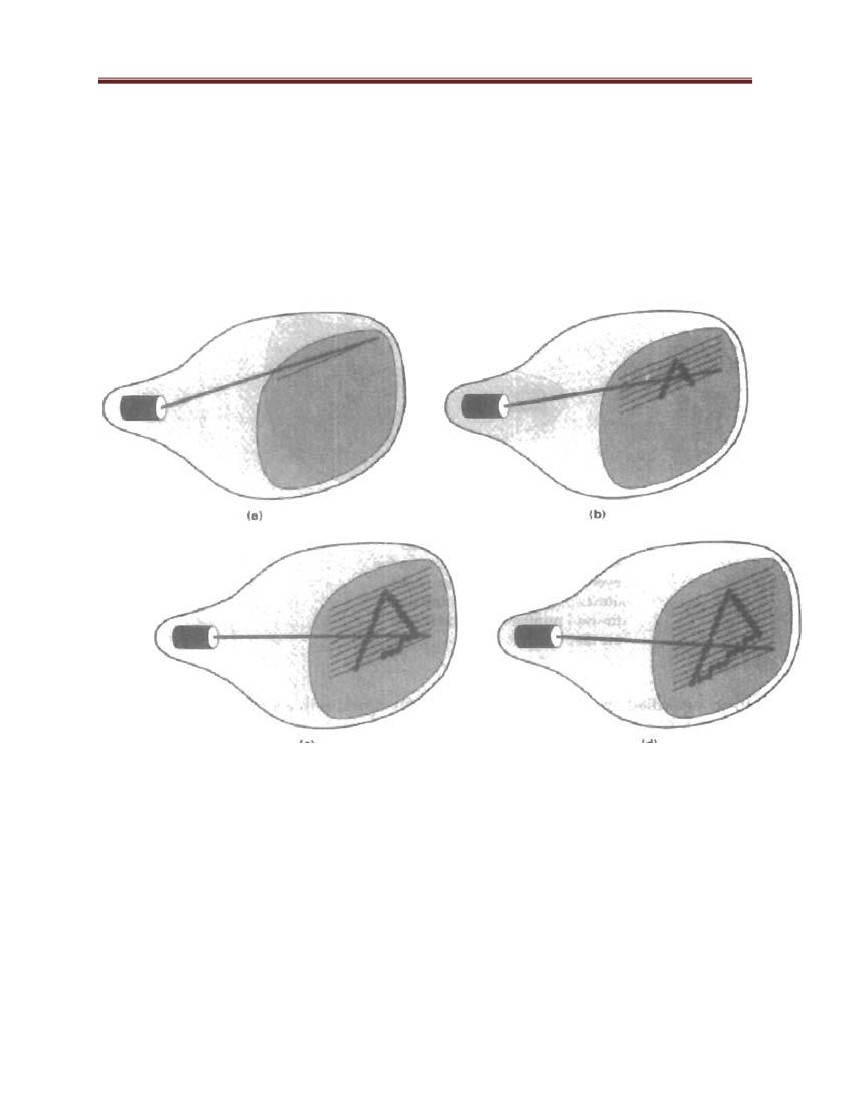
black-and-white system, each screen point is either on or off, so only one bit per pixel is needed

to control the intensity of screen positions. For a bi-level system, a bit value of 1 indicates that

the electron beam is to be turn on at that position, and a value of 0 indicates that the beam

intensity is to be off.

24

Computer Graphics and Vision

Additional bits are needed when color and intensity variations can be displayed. Up to 24 bits per

pixel are included in high-quality systems, which can require several megabytes of storage for

the frame buffer, depending on the resolution of the system. A system with 24 bits per pixel and

a screen resolution of 1024 by 1024 requires 3 megabytes of storage for the frame buffer. On a

black-and-white system with one bit per pixel, the frame buffer is commonly called a bitmap.

For systems with multiple bits per pixel, the frame buffer is often referred to as a pixmap.

Figure 14: A raster-scan system displays an object as a set of points across each scan line.

On some raster-scan systems (and in TV sets), each frame is displayed in two passes using an

interlaced refresh procedure. In the first pass, the beam sweeps across every other scan line from

top to bottom. Then after the vertical retrace, the beam sweeps out the remaining scan lines

(Figure 15). Interlacing of the scan lines in this way allows us to see the entire screen displayed

in one-half the time it would have taken to sweep across all the lines at once from top to bottom.

25

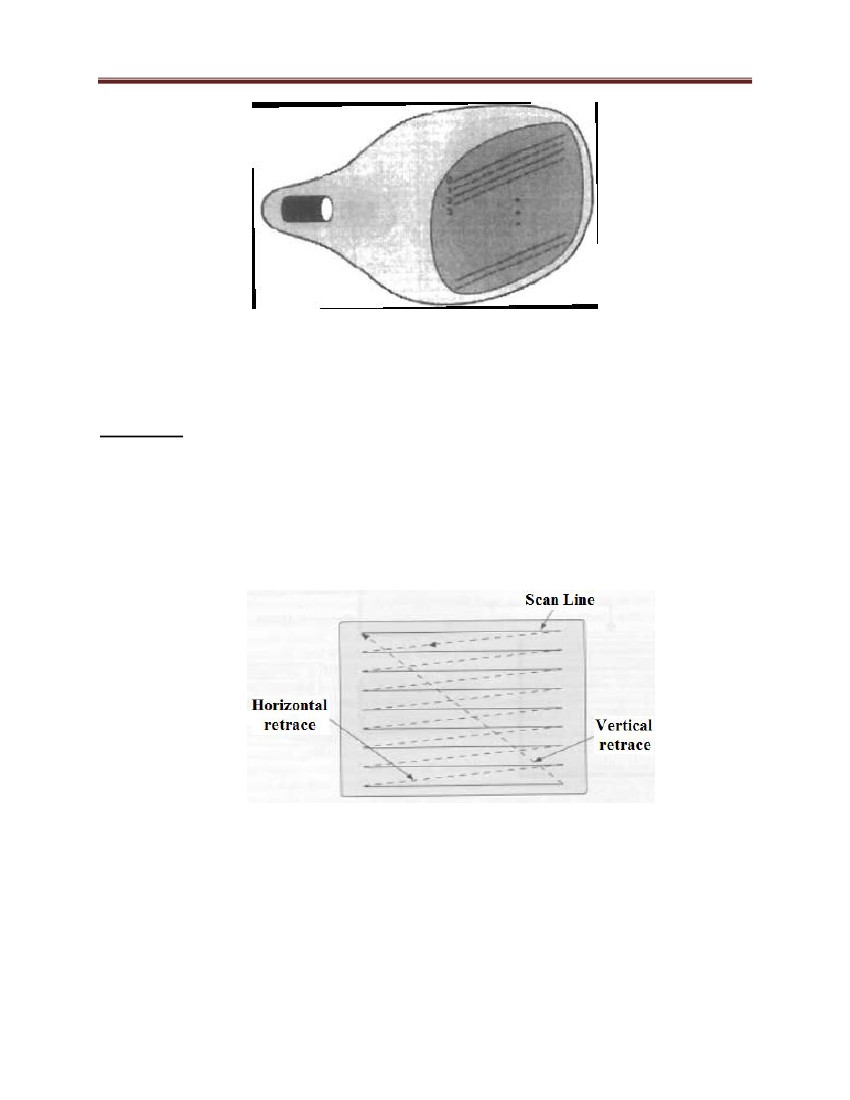
Computer Graphics and Vision

Figure 15 Raster scan display.

First, all points on the well numbered (solid) scan lines are displayed; then all points along the

odd-numbered (dashed) lines are displayed

Raster scan

a) need to store whole image

b) 1024 x1024 x n - n bits per pixel

c) mono 1 bit, color 8 (256 color), 24 (16 million)

d) 32 to 96 bits used (double buffering, z-buffering)

e) 1280x1024x24 needs only 3.75 MB video RAM

Figure 16: Raster Scan

2.3 Raster graphic Devices

Graphics in which an image is generated by scanning an entire screen or page and

marking every point as black, white, or another color, as opposed to vector graphics .

26

Computer Graphics and Vision

A video screen and a laser printer are raster graphics devices; a pen plotter is a vector

graphics device because it marks only at specified points on the page.

2.4 Display FILE Structures

The specific format in which an image file is saved is identified by the three letter

extension at the end of the file name. Every format has its own characteristics, advantages

and disadvantages. By defining the file format it may be possible to determine the

number of bits per pixel and additional information.

2.4.1 General Information:

Images that may be used by PC computers are saved in various formats. Different image

file formats are capable of holding different quantities of colors. Each file format will

have a reference to the number of "bits per pixel" that the format is capable of supporting.

1) 1 bit per pixel refers to an image with 2 colors.

2) 4 bits per pixel refers to an image with up to 16 colors.

3) 8 bits per pixel refers to an image with up to 256 colors.

4) 16 bits per pixel refers to an image with up to 32,768 colors.

5) 24 bits per pixel refers to an image with up to 16,777,216 colors.

The BMP, DIB, and RLE files are known as "Device Independent Bitmap" files, or

"DIB's".

2.4.1.1 BMP (Bit-Map)

BMP is the standard MS-Windows raster format. Windows uses a fixed color palette for

BMP files which cannot be changed, as doing so would make the screen and border

colors change too. This means that transferring an image to the BMP format may result in

some color shifts when BMP files are imported into Windows applications.

2.4.1.2 DIB (Device Independent Bitmap)

DIB files are applied mainly in computer multimedia systems. They can also be can be

used as image files in the Windows environment.

DIB-OS/2-RGB format supports 1, 4, 8, 24 bits per pixel - not compressed.

27

Computer Graphics and Vision

DIB-Windows-RGB format supports 1, 4, 8, 24 bits per pixel - not compressed.

2.4.2 Formats Originating From Specific Applications.

2.4.2.1 GIF (Graphics Interchange Format)

There are two GIF file versions: 87a and 89a

Both versions may use an encoding method referred to as interlacing. When an image is

saved by using four passes instead of just one, it is called interlacing. On each pass,

certain lines of the image are saved to the file. If the program decoding a GIF file

displays the image as it is decoded, the user will be able to see the four passes of the

decoding cycle.

GIF Files can range from monochrome to 256-color.

2.4.2.2 JPEG (Joint Photographic Experts Group)

JPEG compression economizes on the way data is stored and also identifies and discards

extra data, that is, information beyond what the human eye can see. Because it discards

data, the JPEG algorithm is referred to as "lossy". This means that once an image has

been compressed and then decompressed, it will not be identical to the original image. In

most cases, the difference between the original and compressed version of the image is

indistinguishable.

2.4.2.3 TIFF (Tagged-Image File Format)

Tagged-Image File Format (TIFF) is used mainly for exchanging documents between different

applications and different computer platforms. The Tagged Image File Format was primarily

designed to become the standard format. The result of this design provided the flexibility of an

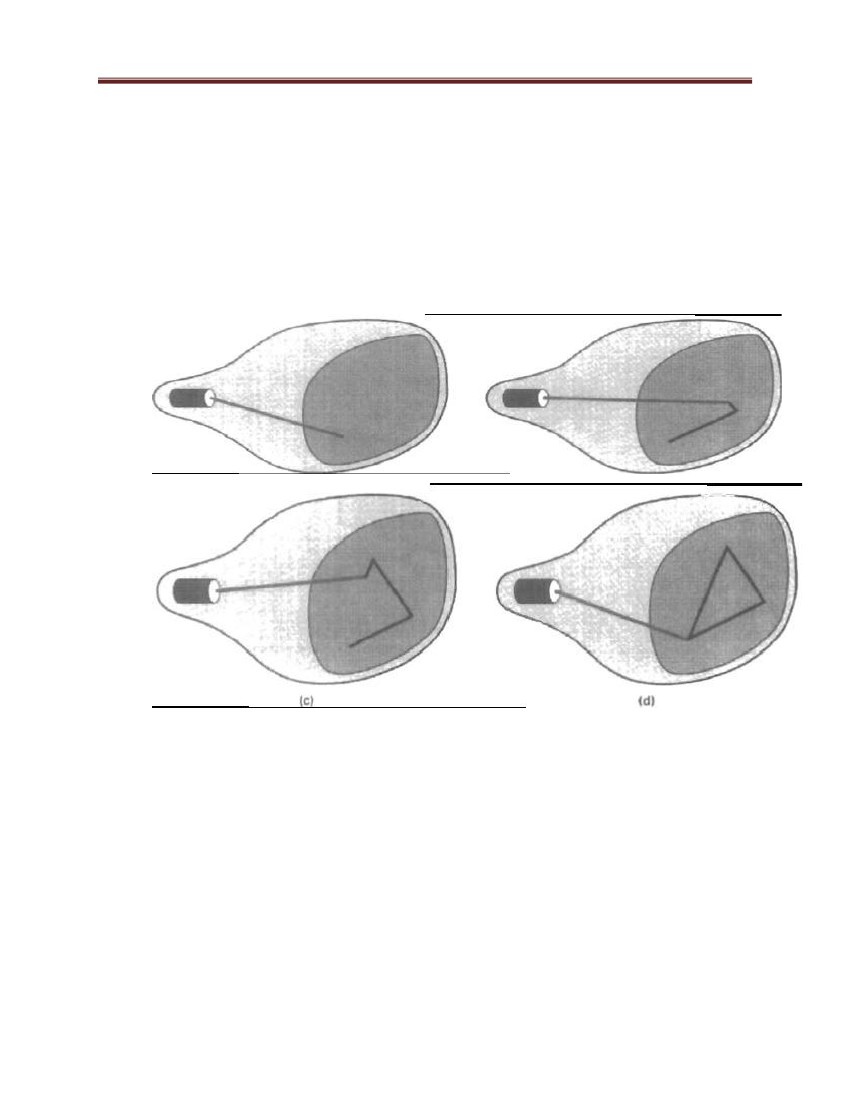
infinite number of possibilities of how a TIFF image can be saved. The TIFF format uses 6

different encoding routines: - No-compression - Huffman - Pack Bits - LZW - Fax Group 3 - Fax

Group 4 In addition it differentiates between types of images in 3 different categories: - Black

and white - Gray scaled - Colored

28

Computer Graphics and Vision

2.5 Random-Scan Displays

When operated as a random-scan display unit, a CRT has the electron beam directed only to the

parts of the screen where a picture is to be drawn. Random scan monitors draw a picture one line

at a time and for this reason are also referred to as vector displays. The component lines of a

picture can be drawn and refreshed by a random-scan system in any specified order (Figure 17).

A pen plotter operates in a similar way and is an example of a random-scan, hard-copy device.

Figure 17: A random-scan system draws the component lines of an object in any order

specified.

2.5.1 Flat-Panel Displays

Although most graphics monitors are still constructed with CRTs, other technologies are

emerging that may soon replace CRT monitors. The term Flat-panel display refers to a class of

video devices that have reduced volume, weight, and power requirements compared to a CRT. A

significant feature of flat-panel displays is that they are thinner than CRTs, and we can hang

them on walls o. Since we can even write on some flat-panel displays, they will soon be

available as pocket notepads. Current uses for flat-panel displays include small TV monitors,

29

Computer Graphics and Vision

calculators, pocket video games, laptop computers, armrest viewing of movies on airlines, as

advertisement boards in elevators, and as graphics displays in applications requiring rugged,

portable monitors.

We can separate flat-panel displays into two categories: emissive displays and non-emissive

displays.

The emissive displays (or emitters) are devices that convert electrical energy into light. Plasma

panels, thin-film electroluminescent displays, and Light-emitting diodes are examples of

emissive displays.

Non-emissive displays (or non-emitters) use optical effects to convert sunlight or light from

some other source into graphics patterns. The most important example of a non-emissive flat-

panel display is a liquid-crystal device.

A type of emissive device is the light-emitting diode (LED). A matrix of diodes is arranged to

form the pixel positions in the display, and picture definition is stored in a refresh buffer. As in

scan-line refreshing of a CRT, information is read from the refresh buffer and converted to

voltage levels that are applied to the diodes to produce the light patterns in the display.

The liquid crystal display (LCDS) are commonly used in small systems, such as calculators and

portable, laptop computers . These non-emissive devices produce a picture by passing polarized

light from the surroundings or from an internal light source through a liquid-crystal material that

can be aligned to either block or transmit the light. The term liquid crystal refers to the fact that

these compounds have a crystalline arrangement of molecules, yet they flow like a liquid.

2.6 Raster-scan systems

Interactive raster graphics systems typically employ several processing units. In addition to the

central processing unit, or CPU, a special-purpose processor, called the video controller or

display controller, is used to control the operation of the display device. Organization of a simple

raster system is shown in Fig. 2-25. Here, the frame buffer can be anywhere in the system

memory, and the video controller accesses the frame buffer to refresh the screen. In addition to

the video controller, more sophisticated raster systems employ other processors as coprocessors

and accelerators to implement various graphics operations.

30

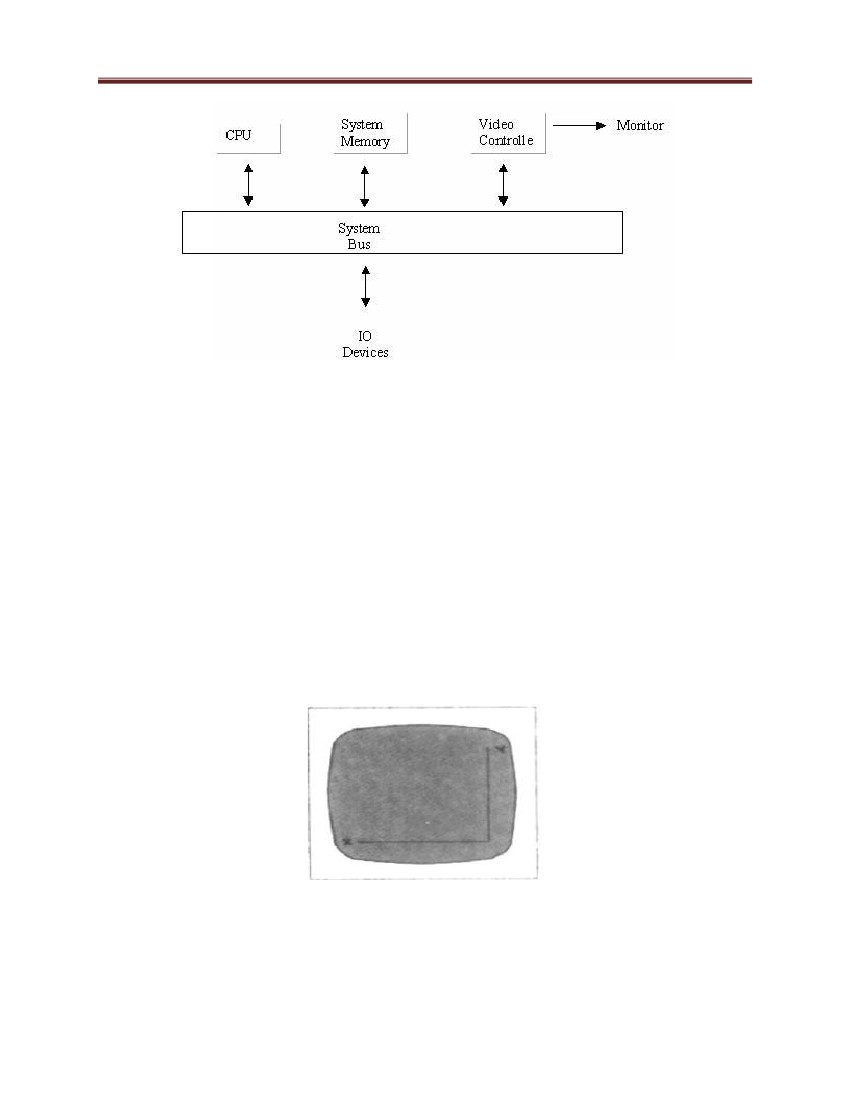
Computer Graphics and Vision

Figure 18: Raster graphics display system

2.6.1 Video Controller

A fixed area of the system memory is reserved for the frame buffer, and the video controller is

given direct access to the frame-buffer memory. Frame-buffer locations, and the corresponding

screen positions, are referenced in Cartesian coordinates. For many graphics monitors, the

coordinate origin is defined at the lower left screen corner. The screen surface is then represented

as the first quadrant of a two-dimensional system, with positive x values increasing to the right

and positive y values increasing from bottom to top. (On some personal computers, the

coordinate origin is referenced to at the upper left comer of the screen, so the y values are

inverted.) Scan lines are then labeled from ymax, at the top of the screen to 0 at the bottom.

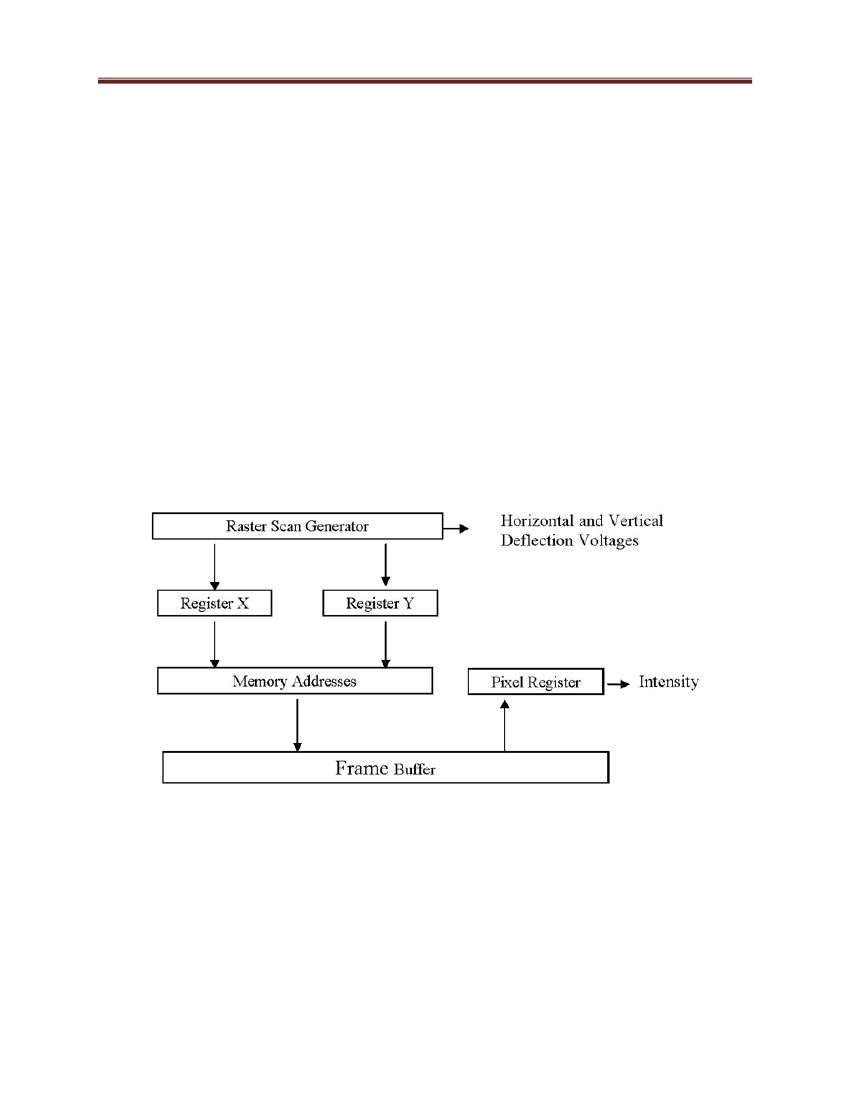
Along each scan line screen pixel positions are labeled from 0 to xmax

Figure 19: Video Controller coordinate

The origin of the coordinate system for identifying screen positions is usually specified in the

lower-left corner

31

Computer Graphics and Vision

A number of other operations can be performed by the video controller, besides the basic

refreshing operations. For various applications, the video controller can retrieve pixel intensities

from different memory areas on different refresh cycles. In high quality systems, for example,

two frame buffers are often provided so that one buffer can be used for refreshing while the other

is being filled with intensity values. Then the two buffers can switch roles. This provides a fast

mechanism-for generating real-time animations, since different views of moving objects can be

successively loaded into the refresh buffers. Also, some transformations can be accomplished by

the video controller. Areas of the screen can be enlarged, reduced, or moved from one location to

another during the refresh cycles. In addition, the video controller often contains a lookup table,

so that pixel values in the frame buffer are used to access the lookup table instead of controlling

the CRT beam intensity directly. This provides a fast method for changing screen intensity

values.

Figure 20: Basic video-controller refresh operations

2.6.2 Raster-Scan Display Processor

The purpose of the display processor is to free the CPU from the graphics chores. In addition to

the system memory, a separate display processor memory area can also be provided.

A major task of the display processor is digitizing a picture definition given in an application

program into a set of pixel-intensity values for storage in the frame buffer. This digitization

32

Computer Graphics and Vision

process is called scan conversion. Graphics commands specifying straight lines and other

geometric objects are scan converted into a set of discrete intensity points. Scan converting a

straight-line segment, for example, means that we have to locate the pixel positions closest to the

line path positions and store the intensity for each position in the frame buffer. Similar methods

are used for scan converting curved lines and polygon outlines. Characters can be defined with

rectangular grids, as in Figure 20. The array size for character grids can vary from about 5 by 7

to 9 by 12 or more for higher-quality displays. A character grid is displayed by superimposing

the rectangular grid pattern into the frame buffer at a specified coordinate position. With

characters that are defined as curve outlines, character shapes are scan converted into the frame

buffer

Scan conversion is the process of converting basic, low level objects into their corresponding

pixel map representations. This is often an approximation to the object, since the frame buffer is

a discrete grid.

Figure 21: A character defined as a rectangular grid of pixel position

Display processors are also designed to perform a number of additional operations.

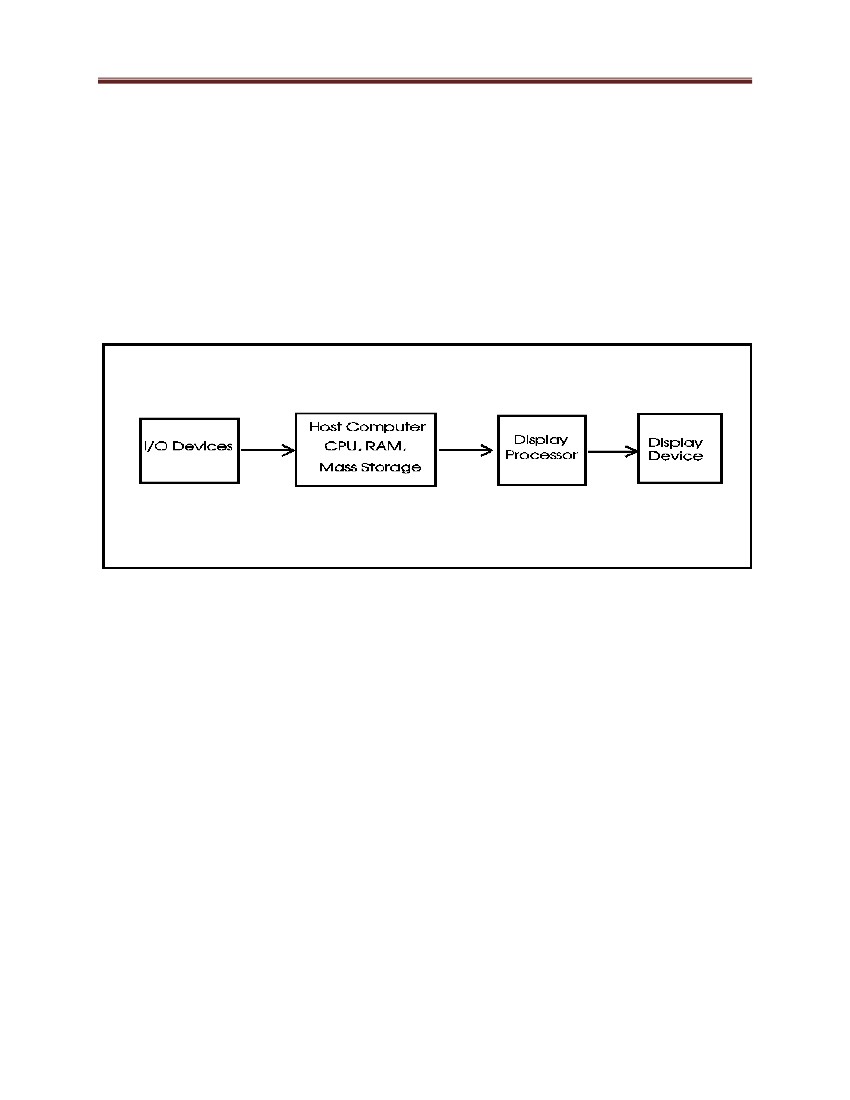
These functions include generating various line styles (dashed, dotted, or solid), displaying color

areas, and performing certain transformations and manipulations on displayed objects. Also,

display processors are typically designed to interface with interactive input devices, such as a

mouse.

33

Computer Graphics and Vision

2.7 Random-scan systems

The organization of a simple random-scan (vector) system is shown in Figure 21. An application

program is input and stored in the system memory along with a graphics package. Graphics

commands in the application program are translated by the graphics package into a display file

stored in the system memory. This display file is then accessed by the display processor to

refresh the screen. The display processor cycles through each command in the display file

program once during every refresh cycle. Sometimes the display processor in a random-scan

system is referred to as a display processing unit or a graphics controller.

Figure 22: A vector random-scan system

Graphics patterns are drawn on a random-scan system by directing the section

electron beam along the component lines of the picture. Lines are defined by the values for their

coordinate endpoints, and these input coordinate values are converted to x and y direction

voltages. A scene is then drawn one line at a time by positioning the beam to fill in the line

between specified endpoints.

3. Graphics Monitors and Workstations

Most graphics monitors today operate as raster scan displays, and here we survey a few of the

many graphics hardware configurations available. Graphics systems range from small general-

purpose computer systems with graphics capabilities to sophisticated full color systems that are

designed specifically for graphics applications. A typical screen resolution for personal computer

systems is 640 by 480, although screen resolution and other system capabilities vary depending

34

Computer Graphics and Vision

on the size and cost of the system. Diagonal screen dimensions for general-purpose personal

computer systems can range from 12 to 21 inches, and allowable color selections range from 16

to over 32,000. For workstations specifically designed for graphics applications, typical screen

resolution is 1280 by 1024, with a screen diagonal of 16 inches or more. Graphics workstations

can be configured with from 8 to 24 bits per pixel (full-color systems), with higher screen

resolutions, faster processors, and other options available in high-end systems.

3.1 Input devices

Various devices are available for data input on graphics workstations. Most systems have a

keyboard and one or more additional devices specially designed for interactive input. These

include a mouse, trackball, space ball, joystick, digitizers, touch panels, image scanners, and

voice systems.

3.1.1 Keyboards

An alphanumeric keyboard on a graphics system is used primarily as a device for entering text

strings. The keyboard is an efficient device for inputting such non graphic data as picture labels

associated with a graphics display. Keyboards can also be provided with features to facilitate

entry of screen coordinates, menu selections, or graphics functions.

Cursor-control keys and function keys are common features on general purpose keyboards.

Function keys allow users to enter frequently used operations in a single keystroke, and cursor-

control keys can be used to select displayed objects or coordinate positions by positioning the

screen cursor. Other types of cursor-positioning devices, such as a trackball or joystick, are

included on some keyboards. Additionally, a numeric keypad is often included on the keyboard

for fast entry of numeric data.

35

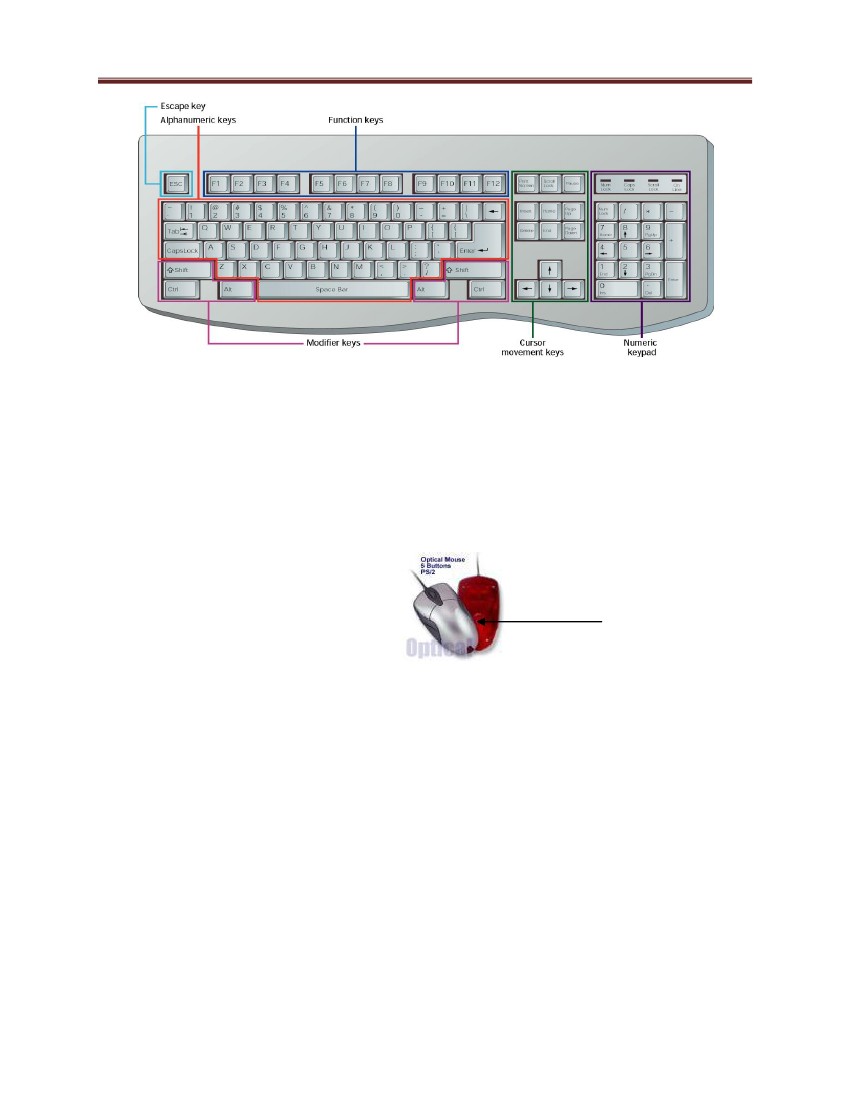
Computer Graphics and Vision

Figure 23: QWERTY Keyboard

3.1.2 Mouse

A mouse is a pointing device that fits comfortably under the palm of your hand. The mouse is

the most widely used pointing device on desktop computers. It is a small handheld device used to

control the movement of a graphical pointer on the screen, often called a mouse pointer in this

case. The mouse can be used to issue commands, draw, and perform other types of input tasks.

Optical sensor

Figure 24: Computer Mouse

3.1.3 Trackball

A trackball is like a mouse turned upside-down. The thumb is used to move the exposed ball and

fingers to press the buttons

As the name implies, a trackball is a ball that can be rotated with the fingers or palm of the hand,

to produce screen-cursor movement.

36

Computer Graphics and Vision

Many styles of trackball are available.

Figure 25: Track ball

3.1.4 Joysticks

A joystick consists of a small, vertical lever (called the stick) mounted on a base that is used to

steer the screen cursor around. Most joysticks select screen positions with actual stick

movement; others respond to pressure on the stick. .

3.1.5 Digitizers

A common device for drawing, painting, or interactively selecting coordinate positions on an

object is a digitizer. These devices can be used to input coordinate values in either a two-

dimensional or a three-dimensional space. Typically, a digitizer is used to scan over a drawing or

object and to input a set of discrete coordinate positions, which can be joined with straight line

segments to approximate the curve or surface shapes. One type of digitizer is the graphics tablet

(also referred to as a data tablet), which is used to input two-dimensional coordinates by

activating a hand cursor or stylus at selected positions on a flat surface.

3.1.6 Image Scanners

Drawings, graphs, color and black-and-white photos, or text can be stored for computer

processing with an image scanner by passing an optical scanning mechanism over the

information to be stored. Some scanners are able to scan either graphical representations or text,

and they come in a variety of sizes and capabilities.

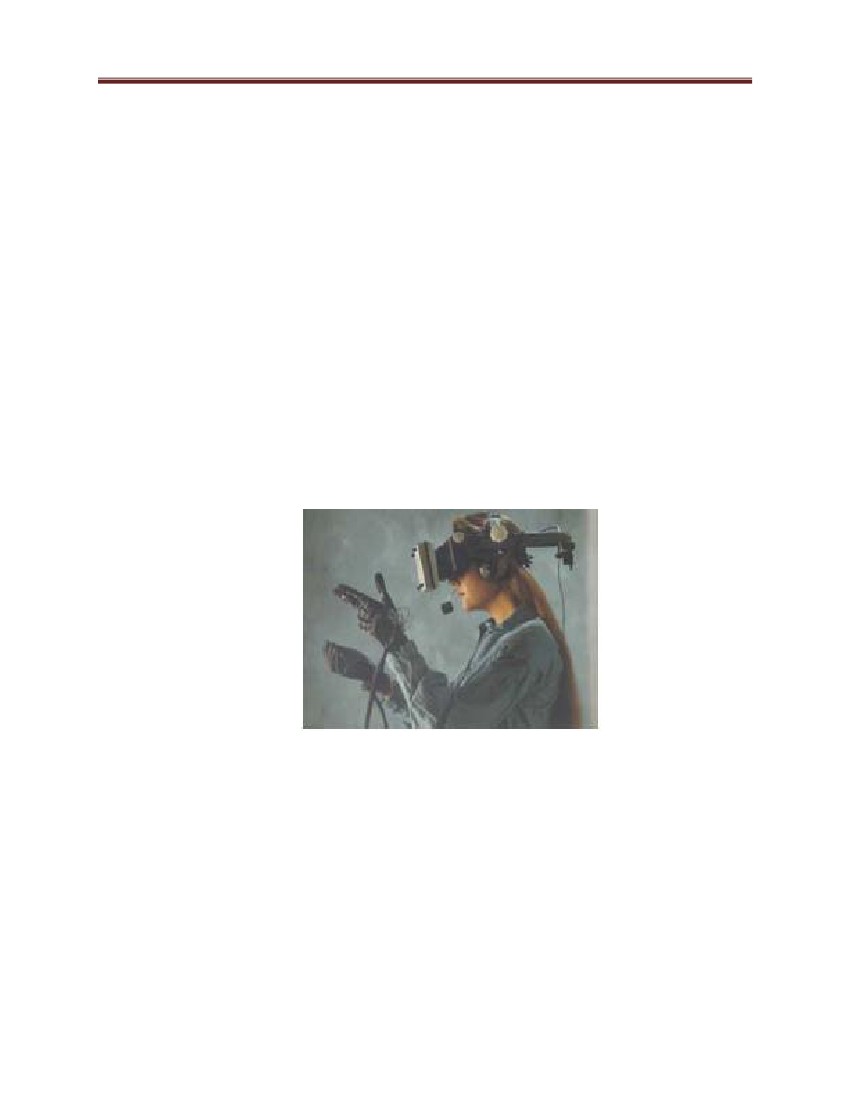
3.1.7 Touch Panels

As the name implies, touch panels allow displayed objects or screen positions to be selected with

the touch of a finger. A typical application of touch panels is for the selection of processing

options that are represented with graphical icons.

37

Computer Graphics and Vision

3.1.8 Light Pens

Such pencil-shaped devices are used to select screen positions by detecting the light coming from

points on the CRT screen. They are sensitive to the short burst of light emitted from the phosphor

coating at the instant the electron beam strikes a particular point.

Pens are common input devices for handheld computers, like ―personal digital assistants

(PDAs).‖ Pens are handy for making notes or selecting commands, not for inputting a lot of text.

3.2 Output devices

•

Stereoscopic viewing glasses: the user wears them to perceive stereoscopic view of

3D scenes displayed on screen

– Used in screen-based Virtual Reality (VR)

– Has high resolution

•

Head-mounted display (HMD): two small TV screens are embedded in a rack and

placed in front of the two eyes.

Figure 26: Illustration of the Head-mounted display

– It allows full-freedom head movement, and gives the feel of immersion

– Widely used in Virtual Reality (VR)

– A tracking system is used to report the position of HMD in 3D space.

•

Wide Screen

38

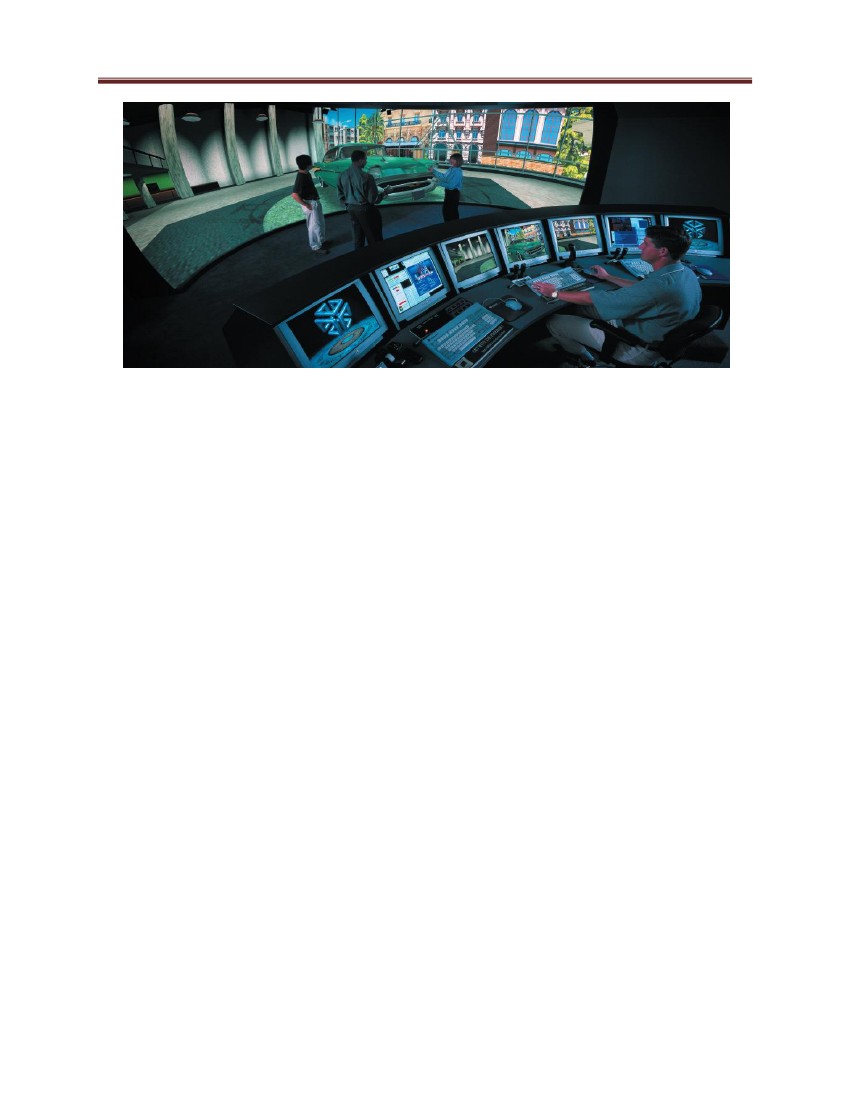
Computer Graphics and Vision

Figure 27: Wide Screen

3.3 Image Printing Devices

We can obtain hard-copy output for our images in several formats. For presentations or

archiving, we can send image files to devices or service bureaus that will produce 35-mm slides

or overhead transparencies. To put images on film, we can simply photograph a scene displayed

on a video monitor. And we can put our pictures on paper by directing graphics output to a

printer or plotter. The quality of the pictures obtained from a device depends on dot size and the

number of dots per inch, or Lines per inch, that can be displayed. To produce smooth characters

in printed text strings, higher-quality printers shift dot positions so that adjacent dots overlap.

Printers produce output by either impact or nonimpact methods. Impact printers press formed

character faces against an inked ribbon onto the paper. A line printer is an example of an impact

device, with the typefaces mounted on bands, chains, drums, or wheels. Nonimpact printers and

plotters use laser techniques, ink-jet sprays, xerographic processes.

Character impact printers often have a dot-matrix print head containing a rectangular array of

protruding wire pins, with the number of pins depending on the quality of the printer. Individual

characters or graphics patterns are obtained by extracting certain pins so that the remaining pins

form the pattern to be printed.

In a laser device, a laser beam mates a charge distribution on a rotating drum coated with a

photoelectric material, such as selenium. Toner is applied to the drum and then transferred to

paper.

39

Computer Graphics and Vision

Ink-jet methods produce output by squirting ink in horizontal rows across a

roll of paper wrapped on a drum. The electrically charged ink stream is deflected

by an electric field to produce dot-matrix patterns.

3.4 Coordinate Representations

With few exceptions, general graphics packages are designed to be used with Cartesian

coordinate specifications. If coordinate values for a picture are specified in some other reference

frame (spherical, hyperbolic, etc.), they must be converted to Cartesian coordinates before they

can be input to the graphics package.

Special-purpose packages may allow use of other coordinate frames that are appropriate

to the application. In general; several different Cartesian reference frames are used to construct

and display a scene.

We can construct the shape of individual objects, such as trees or furniture, in a scene within

separate coordinate reference frames called modeling coordinates, or sometimes local

coordinates or master coordinates. Once individual object shapes have been specified, we can

place the object into appropriate positions within the scene using a reference frame called world

coordinates. Finally, the world-coordinate description of the scene is transferred to one or more

output-device reference frames for display.

These display coordinate systems are referred to as device coordinates. Or screen coordinates in

the case of a video monitor. Generally, a graphics system first converts world-coordinate

positions to normalized device coordinates, in the range from 0 to 1, before final conversion to

specific device coordinates.

3.5 Graphics Functions

A general-purpose graphics package provides users with a variety of functions for creating and

manipulating pictures. These routines can be categorized according to whether they deal with

output, input, attributes, transformations, viewing, or general control. The basic building blocks

for pictures are referred to as output primitives. They include character strings and geometric

entities, such as points, straight lines, curved Lines, filled areas (polygons, circles, etc.), and

shapes defined with arrays of color points. Routines for generating output primitives provide the

basic tools for constructing pictures. Attributes are the properties of the output primitives; that is,

40



Computer Graphics and Vision

an attribute describes how a particular primitive is to be displayed. They include intensity and

color specifications, line styles, text styles, and area-filling patterns. Functions within this

category can be used to set attributes for an individual primitive class or for groups of output

primitives. We can change the size, position, or orientation of an object within a scene using

geometric transformations. The Graphics class defines methods for drawing the following kinds

of shapes:





Lines (drawLine(), which draws a line in the Graphics object's current color, which is

initialized to the Component's foreground color)

Rectangles (drawRect(), fillRect(), and clearRect() -- where fillRect() fills a rectangle with

the Graphics object's current color, and clearRect() fills a rectangle with the Component's

background color)









Round-edged rectangles (drawRoundRect() and fillRoundRect())

Ovals (drawOval() and fillOval())

Arcs (drawArc() and fillArc())

Polygons (drawPolygon() and fillPolygon())

4. Graphic primitives

In Raster Scan display device, the picture information is stored in the frame buffer, the concept

of frame buffer conveys that the information for the image to be projected on the screen is stored

in the form of 0s and 1s, making respective pixels activate and deactivate on the screen, and it is

the concept itself which contributes to the discreteness in the picture under display

4.1 Points

Point plotting is accomplished by converting a single coordinate position furnished by an

application program into appropriate operations for [the output device in use. With a CRT

monitor, for example, the electron beam is turned on to illuminate the screen phosphor at the

selected location. How the electron beam is positioned depends on the display technology. A

random-scan (vector) system stores point-plotting instructions in the display list, and coordinate

values in these instructions are converted to deflection voltages that position the electron beam at

the screen locations to be plotted during each refresh cycle. For a black and white raster system,

41



Computer Graphics and Vision

on the other hand, a point is plotted by setting the bit value corresponding to a specified screen

position within the frame buffer to 1. Then, as the electron beam sweeps across each horizontal

scan line, it emits a burst of electrons (plots a point) whenever a value of 1 is uncounted in the

frame buffer. With an RGB system, the frame buffer is loaded with the color codes for the

intensities that are to be displayed at the pixel positions C program to draw a point here

4.2 Lines

Line drawing is accomplished by calculating intermediate positions along the line path

between two specified endpoint positions. An output device is then directed to fill in

these positions between the endpoints. For analog devices, such as a vector pen plotter or

a random-scan display, a straight line can be drawn smoothly from one endpoint to the

other. Linearly varying horizontal and vertical deflection voltages are generated that are

proportional to the required changes in the x and y directions to produce the smooth line.

Digital devices display a straight line segment by plotting discrete points between the two

endpoints. Discrete coordinate positions along the line path are calculated from the

equation of the line. For a raster video display, the line color (intensity) is then loaded

into the frame buffer at the corresponding pixel coordinates. Reading from the frame

buffer, the video controller then "plots" the screen pixels. To load a specified color into

the frame buffer at a position corresponding to column x along scan line y, we will

assume we have available a low-level procedure

of the form.

setPixel( x,y)

We sometimes will also want to be able to retrieve the current frame buffer intensity

setting for a specified location. We accomplish this with the low-level function:

getpixel (x, y )

In order to draw primitive objects, one has to first scan the converted objects. This refers to the

operation of finding out the location of pixels to be intensified and then setting the values of

corresponding bits, to the desired intensity level. Each pixel on the display surface has a finite

size depending on the screen resolution and hence, a pixel cannot represent a single

42



Computer Graphics and Vision

mathematical point. However, we consider each pixel as a unit square area identified by the

coordinate of its lower left corner, the origin of the reference coordinate system being located at

the lower left corner of the display surface. Thus, each pixel is accessed by a non-negative

integer coordinate pair (x, y). The x values start at the origin and increase from left to right along

a scan line and the y values i.e., the scan line numbers starts at bottom and increase upwards.

Figure 28: Square pixels on the display surface

Figure 28 shows the Array of square pixels on the display surface. Coordinate of pixel A: 0, 0; B:

2, 2; C: 6, 7. A coordinate position (6.26, 7.25) is represented by C, whereas (2.3, 2.5) is

represented by B. Because we need to round off the coordinates to a nearest integer in order to

plot a pixel on the screen, further, we need to say that, it is this rounding off, which leads to

distortion of any graphic image. Line drawing is accomplished by calculating the intermediate

point coordinates along the line path between two given end points. Since, screen pixels are

referred with integer values, plotted positions may only approximate the calculated coordinates –

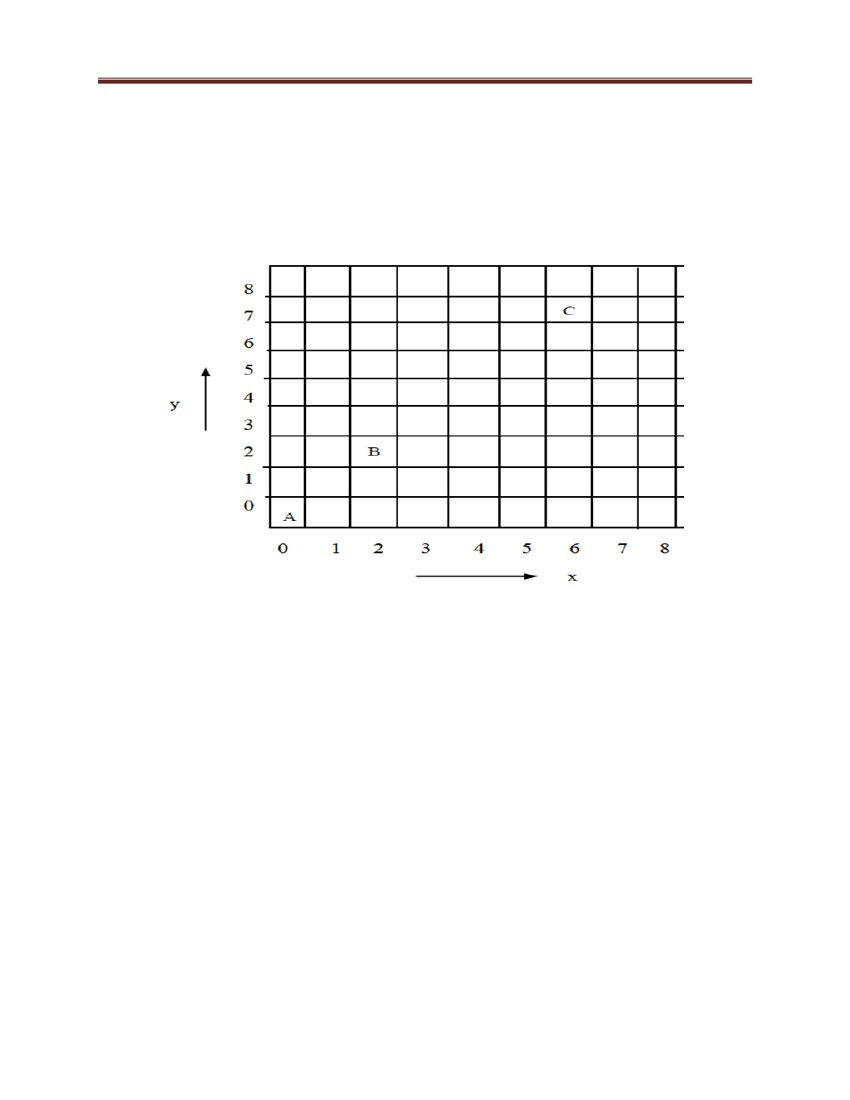
i.e., pixels which are intensified are those which lie very close to the line path if not exactly on

the line path which is in the case of perfectly horizontal, vertical or 45° lines only. Standard

algorithms are available to determine which pixels provide the best approximation to the desired

line, we will discuss such algorithms in our next section. Screen resolution however, is a big

43



Computer Graphics and Vision

factor towards improving the approximation. In a high resolution system the adjacent pixels are

so closely spaced that the approximated line-pixels lie very close to the actual line path and

hence, the plotted lines appear to be much smoother — almost like straight lines drawn on paper.

In a low resolution system, the same approximation technique causes lines to be displayed with a

―stair-step appearance‖ i.e., not smooth as shown in Figure 2, the effect is known as the stair case

effect. We will discuss this effect and the reason behind this defect in the next section of this

unit.

Figure 29: Stair case effect

4.3 Line generation algorithms

We have discussed the case of frame buffer where information about the image to be projected

on the screen is stored in an m x n matrix, in the form of 0s and 1s; the 1s stored in an m\* n

matrix positions are brightened on the screen and 0‘s are not brightened on the screen and this

section which may or may not be brightened is known as the Pixel (picture element). This

information of 0s and 1s gives the required pattern on the output screen i.e., for display of

information. In such a buffer, the screen is also in the form of m\* n matrix , where each section

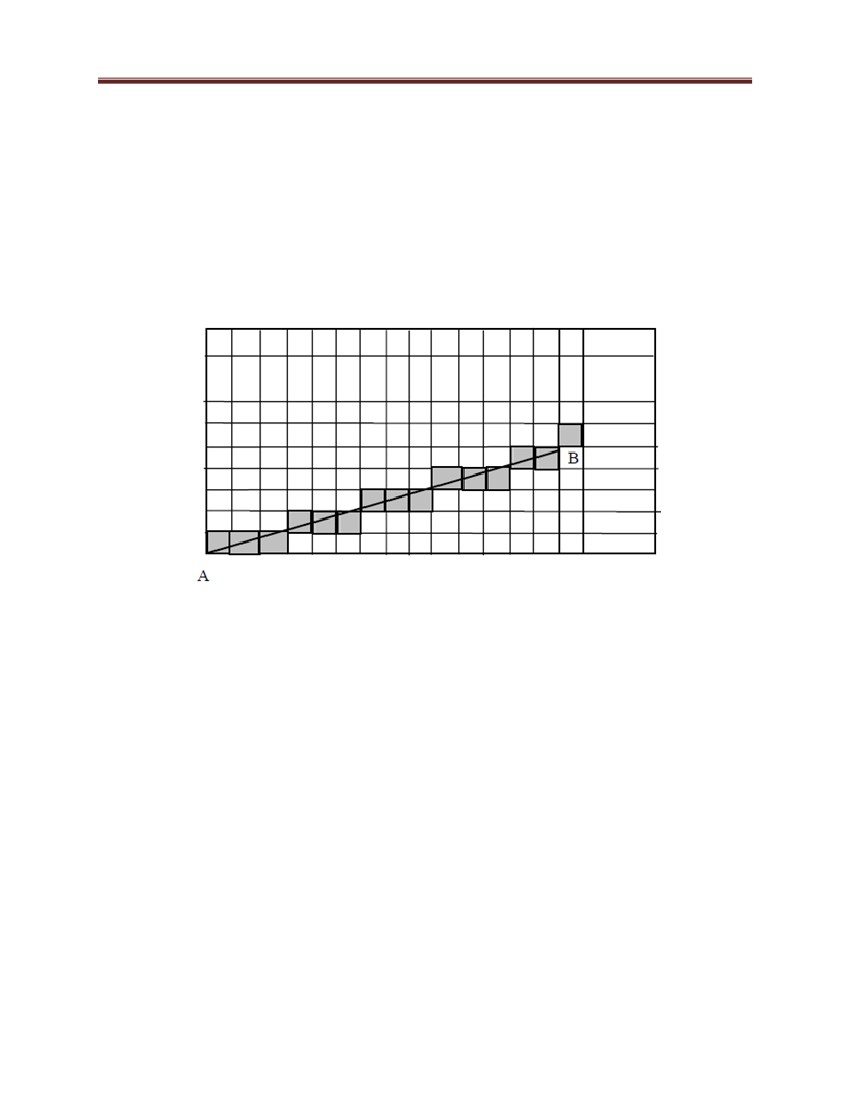
or niche is a pixel (i.e., we have m\* n pixels to constitute the output). The basic idea behind all

the drawing algorithms is to reduce the computations and provide the results rapidly. The

Cartesian slope-intercept equation for a straight line is y m.x b

with m representing the slope of the line and b as they intercept.

44



Computer Graphics and Vision

Given that the two endpoints of a segment are specified at positions (x1, y1,) and (x2, y2),

(X2,y2)

(x1,y1)

Figure 30: The straight line in Cartesian coordinates

we can determine values for the slope m and y intercept b with the following calculations:

m

y 2 y1

x 2 x1

The Cartesian slop-intercept equation for a straight line is

y= mx+b

with:

m, the slope

The 2 end points of a line segment are specified at a position(x1,y1)

Determine the values for the slope m and y intercept b with the following calculation.

here, slope m:

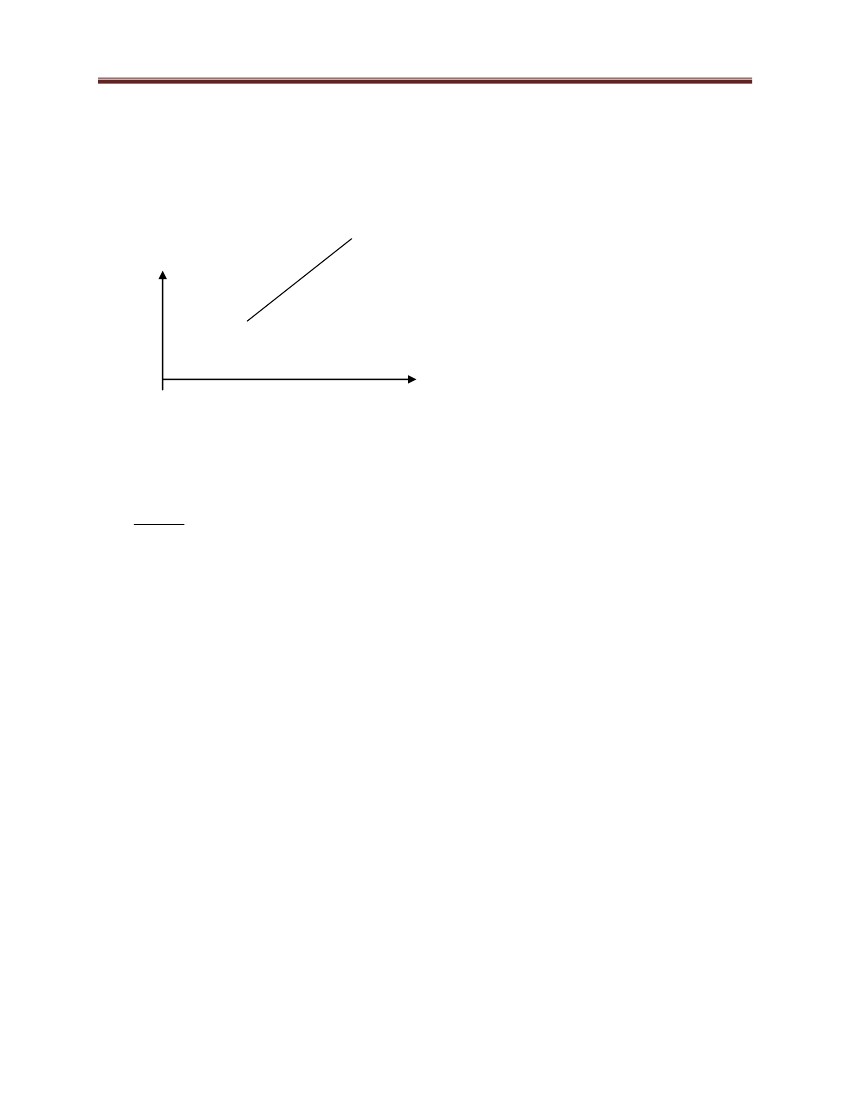
m = ( y2 - y1) / ( x2 - x1 )

m= Dy/Dx

y intercept b

Algorithms can be structured to display the straight line based on this equation.

45



Computer Graphics and Vision

Figure 31: Basic Line Generation

Now, it is to be noted that the creation of a line is merely not restricted to the above pattern,

because, sometimes the line may have a slope and intercept that its information is required to be

stored in more than one section of the frame buffer, so in order to draw or to approximate such

the line, two or more pixels are to be made ON. Thus, the outcome of the line information in the

frame buffer is displayed as a stair; this effect of having two or more pixels ON to approximating

a line between two points say A and B is known as the Staircase effect. The concept is shown

below in Figure 32.

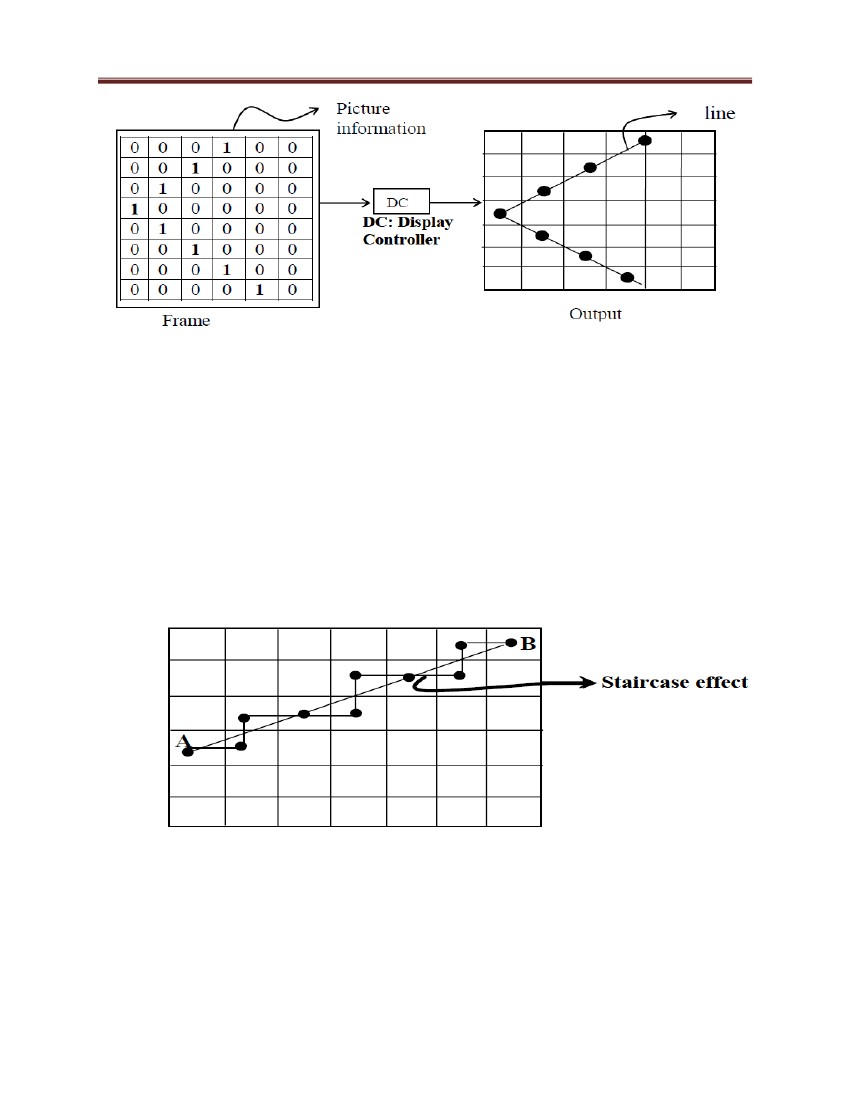
Figure 32: Staircase effect

So, from the Figure 32, it is clear to that when a line to be drawn is simply described by its end

points, then it can be plotted by making close approximations of the pixels which best suit the

line, and this approximation is responsible for the staircase effect, which miss projects the

46



Computer Graphics and Vision

information of the geometry of line stored in the frame buffer as a stair. This defect known as

Staircase effect is prominent in DDA Line generation algorithms, thus, to remove the defect

Bresenham line generation Algorithm was introduced. We are going to discuss DDA (Digital

Differential Analyzer) Algorithm and Bresenham line generation Algorithm a little bit later.

4.3.1 DDA (Digital Differential Analyzer) Algorithm

From the above discussion we know that a Line drawing is accomplished by calculating

intermediate point coordinates along the line path between two given end points. Since screen

pixels are referred with integer values, or plotted positions, which may only approximate the

calculated coordinates – i.e., pixels which are intensified are those which lie very close to the

line path if not exactly on the line path which in this case are perfectly horizontal, vertical or 45°

lines only. Standard algorithms are available to determine which pixels provide the best

approximation to the desired line, one such algorithm is the DDA (Digital Differential Analyzer)

algorithm. Before going to the details of the algorithm, let us discuss some general appearances

of the line segment, because the respective appearance decides which pixels are to be intensified.

It is also obvious that only those pixels that lie very close to the line path are to be intensified

because they are the ones which best approximate the line. Apart from the exact situation of the

line path, which in this case are perfectly horizontal, vertical or 45° lines (i.e., slope zero,

infinite, one) only. We may also face a situation where the slope of the line is > 1 or < 1.Which

is the case shown in Figure 33

47



Computer Graphics and Vision

Figure 33: DDA line generation

In Figure 33, there are two lines. Line 1 (slope<1) and line 2 (slope>1). Now let us discuss the

general mechanism of construction of these two lines with the DDA algorithm. As the slope of

the line is a crucial factor in its construction, let us consider the algorithm in two cases depending

on the slope of the line whether it is > 1 or < 1.

Case 1: slope (m) of line is < 1 (i.e., line 1): In this case to plot the line we have to move the

direction of pixel in x by 1 unit every time and then hunt for the pixel value of the y direction

which best suits the line and lighten that pixel in order to plot the line.

So, in Case 1 i.e., 0 < m < 1 where x is to be increased then by 1 unit every time and proper y is

approximated.

If the slope is less than or equal to 1 ,the unit x intervals Dx=1 and compute each successive y

values.

Dx=1

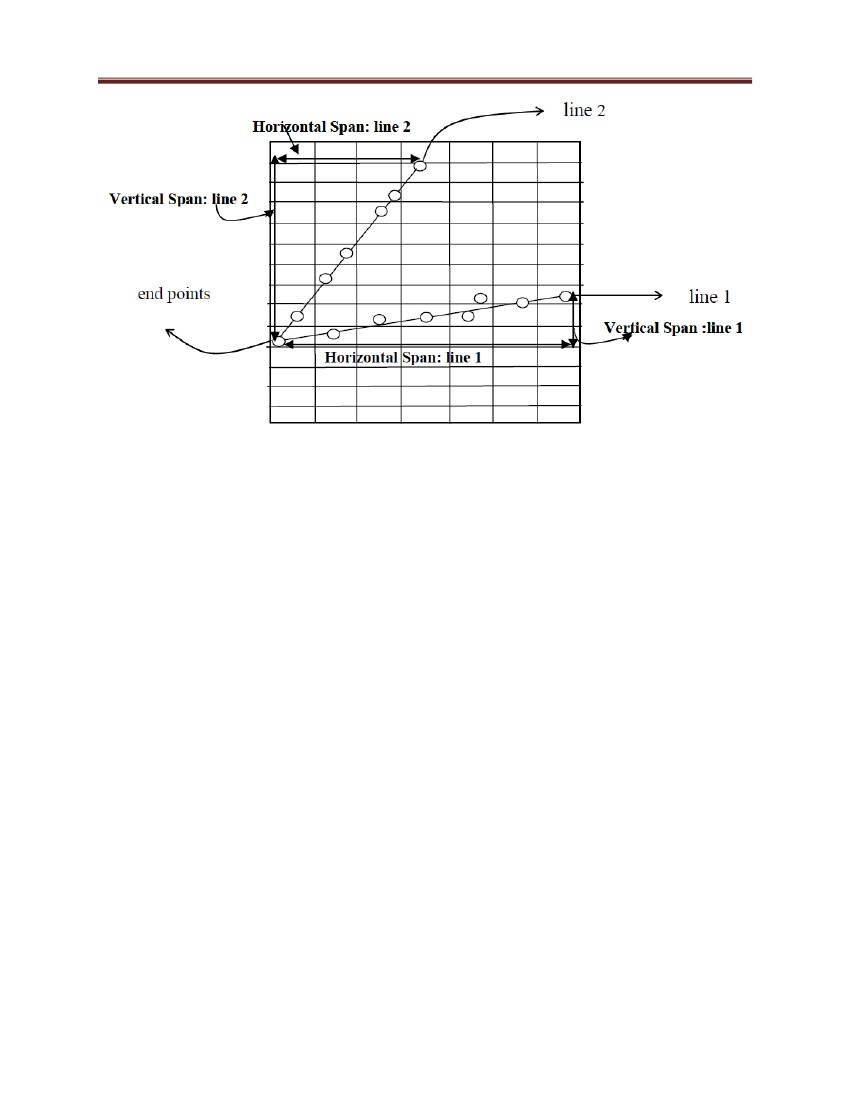
m = Dy / Dx

m = ( y2-y1 ) / 1

m = ( yk+1 – yk ) /1

yk+1 = yk + m

48



Computer Graphics and Vision

Case 2: slope (m) of line is > 1 (i.e., line 2) if m > 1 i.e., case of line 2, then the most

appropriate strategy would be to move towards the y direction by 1 unit every time and

determine the pixel in x direction which best suits the line and get that pixel lightened to plot the

line.

So, in Case 2, i.e., ∞>m>1 or simply 1<m<∞ where y is to be increased by 1 unit every time and

proper x is approximated.

If the slope is greater than 1 ,the roles of x and y at the unit y intervals Dy=1 and compute each

successive y values.

Dy=1

m= Dy / Dx

m= 1/ ( x2-x1 )

m = 1 / ( xk+1 – xk )

xk+1 = xk + ( 1 / m )

st

Assumption: The line generation through DDA is discussed only for the I Quadrant, if the line

lies in any other quadrant then apply respective transformation (generally reflection

st

transformation), such that it lies in I Quadrant and then proceed with the algorithm, also make

intercept Zero by translational transformation such that (xi, yi) resolves to (xi, mxi + c) or (xi,

mxi) and similar simplification occurs for other cases of line generation. The concept and

application of transformations is discussed in Block 2 of this course.

Note:

1) If in case 1, we plot the line the other way round i.e., moving in y direction by 1 unit every

time and then hunting for x direction pixel which best suits the line. In this case, every time

we look for the x pixel, it will provide more than one choice of pixel and thus enhances the

defect of the stair case effect in line generation. Additionally, from the Figure 5, you may

notice that in the other way round strategy for plotting line 1, the vertical span is quite less in

comparison to the horizontal span. Thus, a lesser number of pixels are to be made ON, and

will be available if we increase Y in unit step and approximate X. But more pixels will be

available if we increase X in unit steps and approximate Y (this choice will also reduce

staircase effect distortion in line generation) (more motion is to be made along x-axis).

49

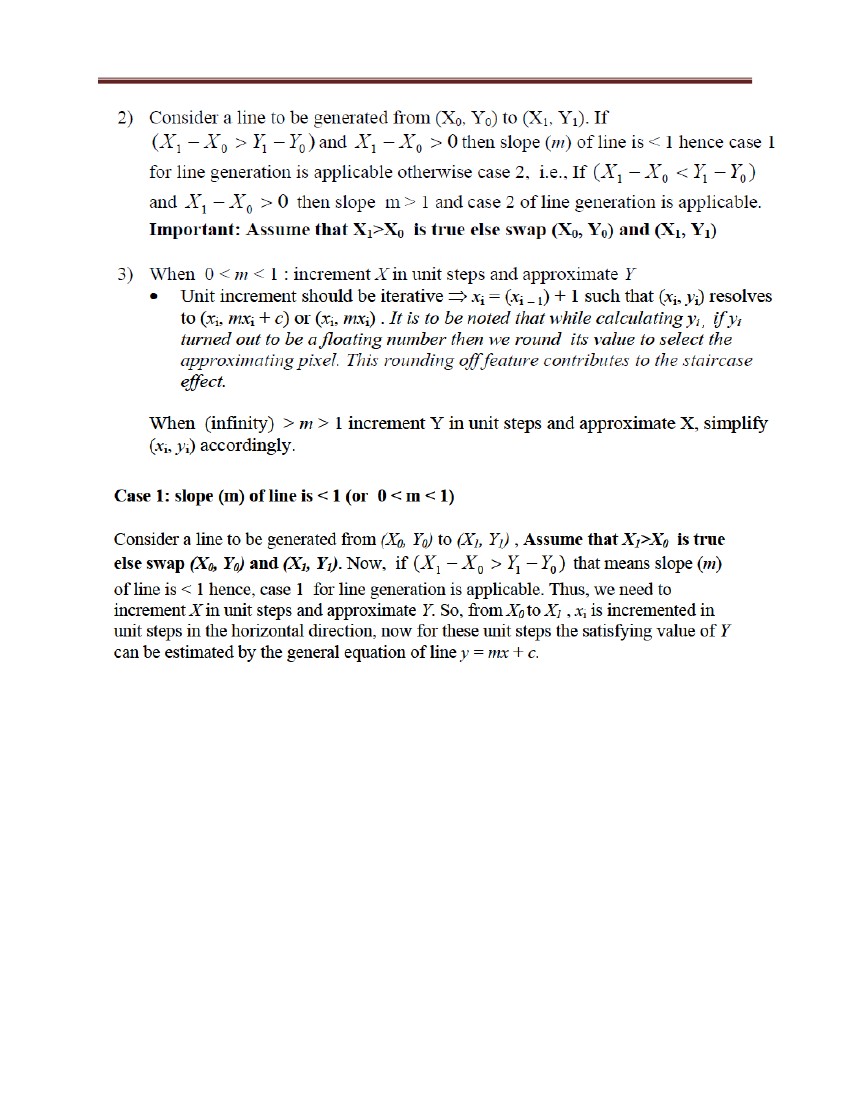


Computer Graphics and Vision

Similarly, for case 2, let us sum up our discussion on DDA algorithm for both

cases. We will examine each case separately.

50



Computer Graphics and Vision

Figure 34 Slope (m) of line is < 1 (i.e., line 1)

Summary:

For instance, in a given line the end points are (X0, Y0) and (X1, Y1). Using these

end points we find the slope (m = Y1 – Y0/ X1 – X0) and check that the value of m

lies between 0 and 1 or is > 1. If 0 < m < 1 then case 1 applies, else, case 2 applies.

For case 1, increase x by one Unit every time, for case 2 increase y by one Unit

every time and approximate respective values of y and x

We assume equation of line is y = mx+c

At x = xi we have yi = mxi+c

Similarly at x = xi + 1 we have yi + 1 = mxi + 1 +c

Case 1: Slope (m) of line is 0 < m1 (i.e., line 1)

Since x is to be increase by 1 unit each time

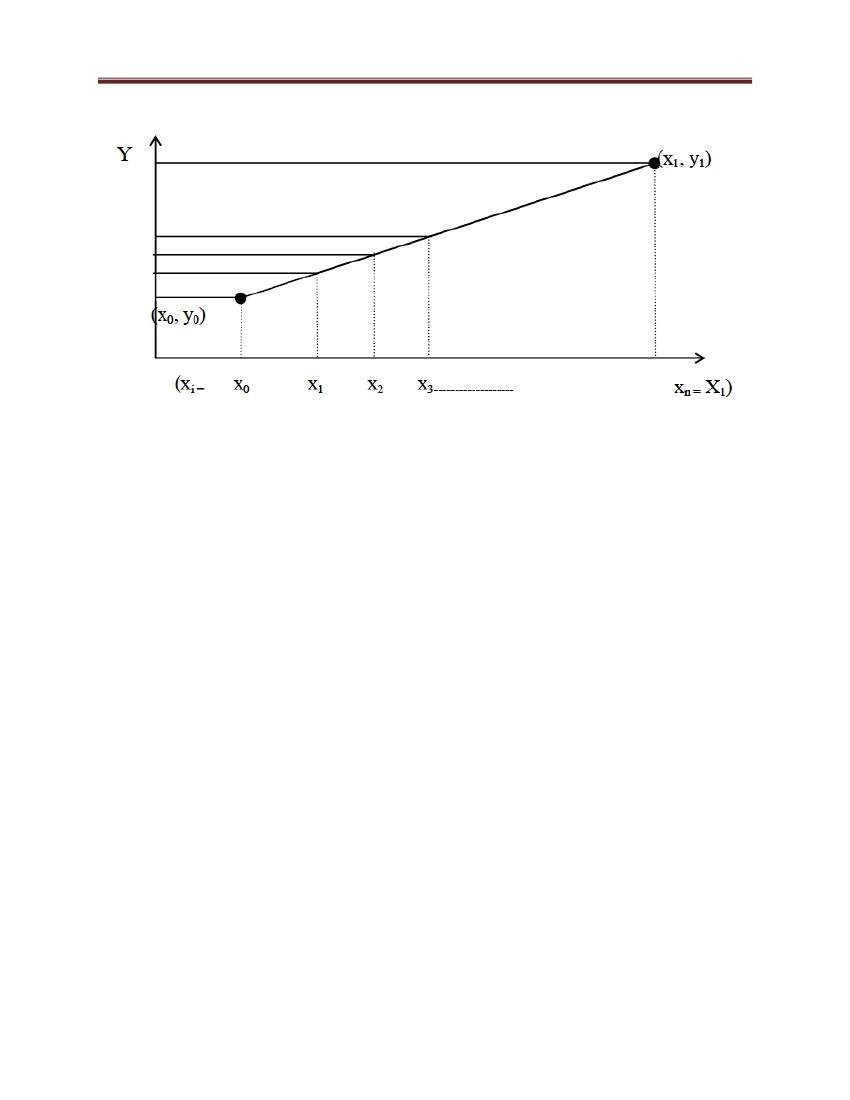
⇒ xi + 1 = xi + 1 ---------------------------------------------------------------------------- (1)

So by using equation of line y = mx+c we have

yi + 1 = m (xi + 1) +c

= mxi +c + m

51



Computer Graphics and Vision

= yi + m ----------------------------------------------------------------------------------------------------- (2)

Equations (1) and (2) imply that to approximate line for case 1 we have to move along x

direction by 1 unit to have next value of x and we have to add slope m to initial y value to get

next value of y.

Now, using the starting point (x0, y0) in the above equations (1) and (2) we go for x i and yi (i = 1,

2, ……n) and put color to the pixel to be lightened.

It is assumed that X0 < X1 ∴the algorithm goes like:

x ← X0

y ← Y0

m ← (Y1 – Y0)/ (X1 – X0)

while (x < = X1) do

{ put-pixel (x, round (y), color)

(new x-value) x ← (old x-value) x + 1

(new y-axis) y ← (old y-value) y + m

}

Sample execution of algorithm case 1:

at (x0, y0) : put-pixel (x0, y0, color)

x1 = x0 + 1; y1 = y0 + m

at (x1, y1) = put pixel (x1, y1, color)

similarly, x2 = x1 + 1; y2 = y1 + m

at (x2, y2) : put pixel (x2, y2, color) and so on.

Case 2: slope (m) of line is > 1 (i.e., line 2): Same as case 1 but, this time, the y

component is increased by one unit each time and appropriate x component is to be

selected. To do the task of appropriate selection of x component we use the equation of

Line: y = mx+c.

Unit increment should be iterative ⇒ yi+1 = yi + 1; for this yi+1 we find corresponding xi+1

by using equation of line y = mx + c and hence get next points (xi+1, yi+1).

52



Computer Graphics and Vision

⇒ yi + 1 – yi = m (xi + 1 – xi) ----------- (3)

as y is to be increase by unit steps

∴yi + 1 – yi = 1 -------------------------- (4)

using equation (4) in (3) we get

⇒ l = m (xi + 1 – xi) -------------------- (5)

rearranging (5) we get

⇒ 1/m = (xi + 1 – xi)

⇒

So, procedure as a whole for Case 2 is summed up as Algorithm case 2:

Assuming Y0 < Y1 the algorithm goes like

Algorithm for case 2:

x ← X0;

y ← Y0; Y

m ← (Y1 – Y0)/ (X1 – X0);

m1 ← 1/m;

while (y < Y1) do

{

put-pixel (round (x), y, color)

y ← y + 1;

x ← x + m1;

xi + 1 = xi + m1

Example 1: Draw line segment from point (2, 4) to (9, 9) using DDA algorithm.

Solution: We know general equation of line is given by

y = mx+c where m =( y1 – y0/( x1 – x0)

given (x0, y0) → (2, 4) ; (x1, y1) → (9, 9)

53



Computer Graphics and Vision

So, by equation of line ( y = mx + c) we have

y = 5/7\*1 8/18

DDA Algorithm Two case:

Case 1: m < 1

xi + 1 = xi + 1

yi + 1 = yi + m

Case 2: m > 1

xi + 1 = xi + (1/m)

yi + 1 = yi + 1

As 0 < m < 1 so according to DDA algorithm case 1

xi + 1 = xi + 1 yi + 1 = yi + m

given (x0, y0) = (2, 4)

1) x1 = x0 + 1 = 3

y1 = y0 + m = 4+5/7=33/7

put pixel (x0, round y, color)

i.e., put on (3, 5)

2) x2 = x1 + 1 = 3 + 1 = 4

y2 = y1 + m = (33/7) + 5/7 = 38/7

put on (4, 5)

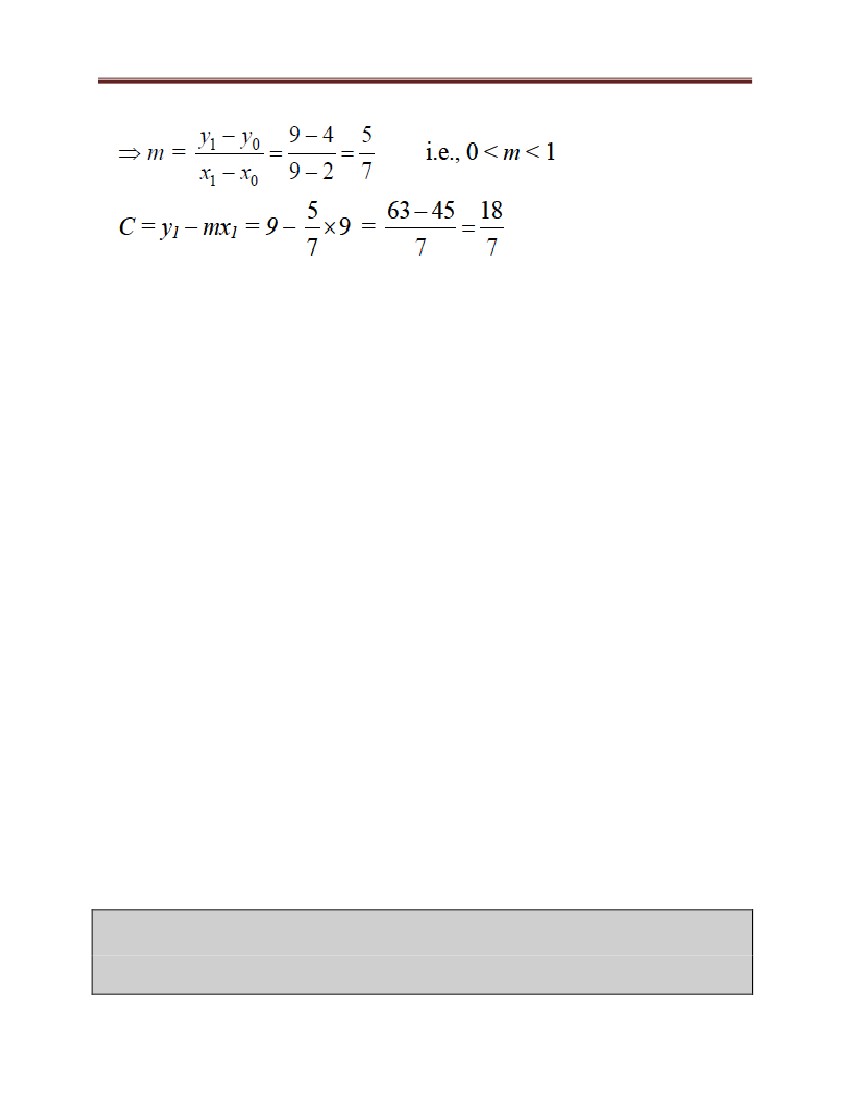
Similarly go on till (9, 9) is reached.

C program to draw a line using DDA (Digital Differential Analyzer) Algorithm

void draw\_line\_dda(int x0, int y0, int x1, int y1)

{

54



Computer Graphics and Vision

int x;

float y;

int dx, dy;

float m;

dx = x1 - x0;

dy = y1 - y0;

m = (float)dy / dx;

y = y0;

for (x = x0; x <= x1; x++)

{

putpixel(x, y + 0.5, color);

y = y + m;

}

}

Example

xa,ya=>(2,2)

xb,yb=>(8,10)

dx=6

dy=8

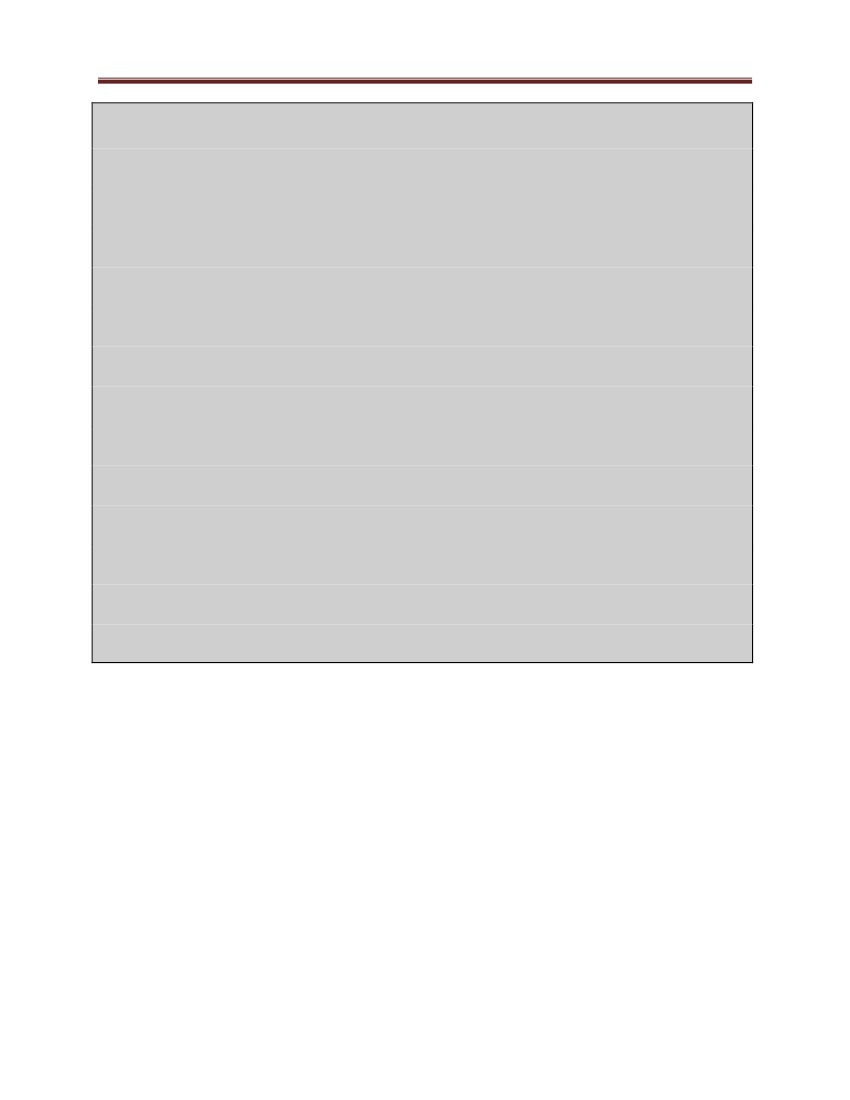
xincrement=6/8=0.75

yincrement=8/8=1

1)

for(k=0;k<8;k++)

55



Computer Graphics and Vision

xincrement=0.75+0.75=1.50

yincrement=1+1=2

1=>(2,2)

2)

for(k=1;k<8;k++)

xincrement=1.50+0.75=2.25

yincrement=2+1=3

2=>(3,3)

it will be incremented up to the final end point is reached

#include < graphics.h >

#include < stdio.h >

#include < conio.h >

#include < math.h >

void main()

{

int gd=DETECT,gm=DETECT,s,dx,dy,m,x1,y1,x2,y2;

float xi,yi,x,y;

clrscr();

printf("Enter the sarting point x1 & y1n");

scanf("%d%d",&x1,&y1);

printf("Enter the end point x2 & y2n");

scanf("%d%d",&x2,&y2);

initgraph(&gd,&gm,"");

cleardevice();

dx=x2-x1;

dy=y2-y1;

if(abs(dx)>abs(dy))

s=abs(dx);

else

s=abs(dy);

xi=dx/(float)s;

56



Computer Graphics and Vision

yi=dy/(float)s;

x=x1;

y=y1;

putpixel(x1+0.5,y1+0.5,4);

for(m=0; m < s; m++)

{

x+=xi;

y+=yi;

putpixel(x+0.5,y+0.5,4);

}

getch();

}

4.3.2 Bresenham Line Generation Algorithm

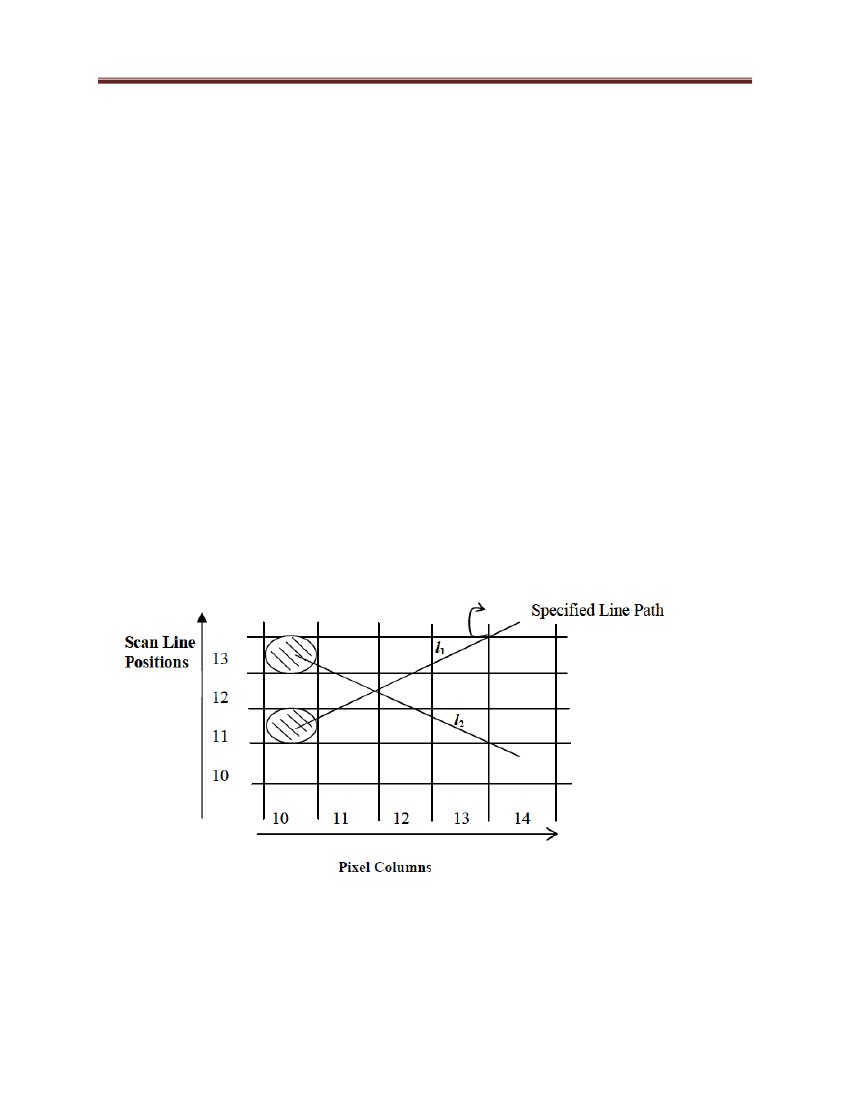
Bresenham algorithm is accurate and efficient raster line generation algorithm. This algorithm

scan converts lines using only incremental integer calculations and these calculations can also be

adopted to display circles and other curves

Figure 35: Bresenham Line Generation process

57



Computer Graphics and Vision

Sampling at Unit x distance in Figure 34, we need to decide which of the two possible

pixel position is closer to the line path at each sample step.

In l1 : we need to decide that at next sample position whether to plot the pixel at position

(11, 11) or at (11, 12).

Similarly, In l2: the next pixel has to be (11, 13) or (11, 12) or what choice of pixel is to

be made to draw a line is given by Bresenham, by testing the sign of the integer

parameter whose value is proportional to the difference between the separation of the two

pixel positions from actual line path. In this section, we will discuss the Bresenham line

drawing algorithm for + ve slope ( 0 < m < 1). If the slope is negative then, use reflection

transformation to transform the line segment with negative slope to line segment with

positive slope. Now, let us discuss the generation of line again in two situations as in the

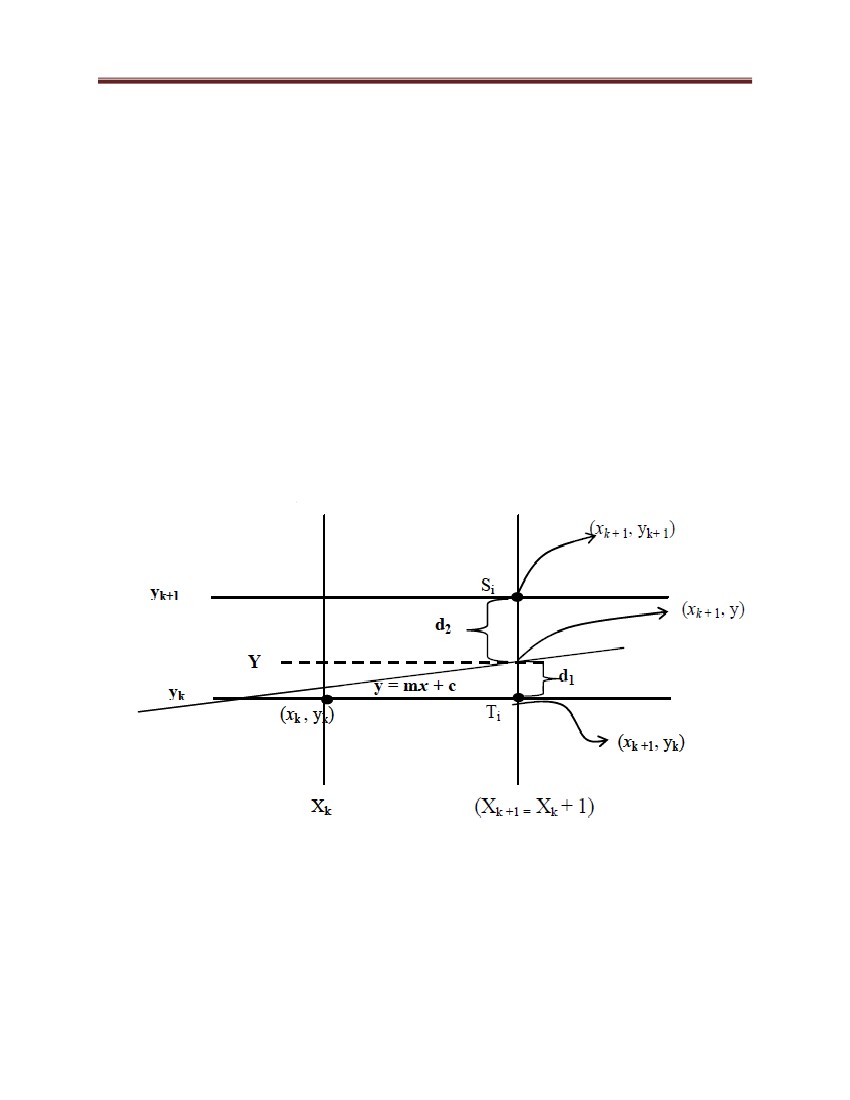
case of DDA line generation

Figure 36: Sampling at x distance

At sampling position Xk + 1 the vertical pixel (or scan line) separation from mathematical line (y =

mx + c) is say d1 and d2.

58



Computer Graphics and Vision

Now, the y coordinate on the mathematical line at pixel column position Xk + 1 is:

y = m (xk + 1) + c ……………………………………………………………………………….. (1)

d1 = y – yk ……………………………………………………………………………...….….... (2)

= m (xk + 1) + c – yk -………………………………………………………………………….… (3)

similarly, d2 = (yk + 1) – y = yk + 1 – m (xk + 1) – c……………………………………………….. (4)

using (3) and (4) we find d1 – d2

d1 – d2 = [m (xk + 1) + c – yk] – [yk + 1 – m (xk + 1) – c]

= mxk + m + c – yk – yk – 1 + mxk + m + c

= 2m (xk + 1) – 2yk + 2c – 1……………………………………………………………..……... (5)

th

Say, decision parameter p for k step i.e., pk is given by

…………………………………………………………………………...… (6)

th

Now, a decision parameter pk for the k step in the line algorithm can be obtained by rearranging

(5) such that it involves only integer calculations. To accomplish this substitute m = Δy/Δx

where, Δy and Δx ⇒ vertical and horizontal separations of the end point positions.

pk = Δx (d1 – d2) = Δ x [2m (xk + 1) – 2yk + 2c – 1]

= Δx [2(Δy/Δx) (xk + 1) – 2yk + 2c – 1]

= 2 Δy (xk + 1) – 2 Δxyk + 2 Δxc – Δx

= 2 Δy xk – 2 Δx yk + [2 Δy + Δx (2c – 1)]

pk = 2 Δy xk – 2Δxyk + b ……………………………………………………… (7)

where b is constant with value b = 2Δy + Δx (2c – 1)………………………………………….(8)

Note: sign of pk is same as sign of d1 – d2 since it is assumed that Δx > 0

[d1 < d2 i.e. (d1 – d2) is –ve i.e., pk is –ve so pixel Ti is more appropriate choice otherwise pixel Si

is the appropriate choice.

i.e., (1) if pk < 0 choose Ti , so next pixel choice after (xk, yk) is (xk + 1, yk)

else (2) if pk > 0 choose Si , so next pixel choice after (xk , yk ) is (xk + 1, yk + 1).

At step k + 1 the decision parameter is evaluated by writing (7) as:

59



Computer Graphics and Vision

pk + 1 = 2Δy xk + 1 – 2Δx yk + 1 + b ………………………………………………………………..(9)

Subtracting (7) from (9) we get

This recursive calculation of decision parameter is preformed at each integer position, beginning

with the left coordinate end point of line.

This first parameter p0 is determined by using equation (7), and (8) at the starting pixel position

(x0, y0) with m evaluated as Δy /Δx (i.e., intercept c = 0)

p0 = 0 – 0 + b = 2Δy + Δx (2 \* 0 – 1) = 2Δy – Δx

p0 = 2Δy – Δx ………………………………………………………………………………… (10)

Algorithm | m | < 1:

(a) Input two line end points and store left end point in (x0, y0)

(b) Load (x0, y0) on frame buffer i.e., plot the first point.

(c) Calculate Δx, Δy, 2Δy, 2Δy – 2Δx and obtain the starting value of decision parameter as p0 =

2Δy – Δx

(d) At each xk along the line, starting at k= 0, perform following test:

if pk < 0, the next plot is (xk + 1, yk) and pk + 1 = pk + 2Δy

else next plot is (xk + 1 , yk + 1) and pk + 1 = pk + 2(Δy – Δx)

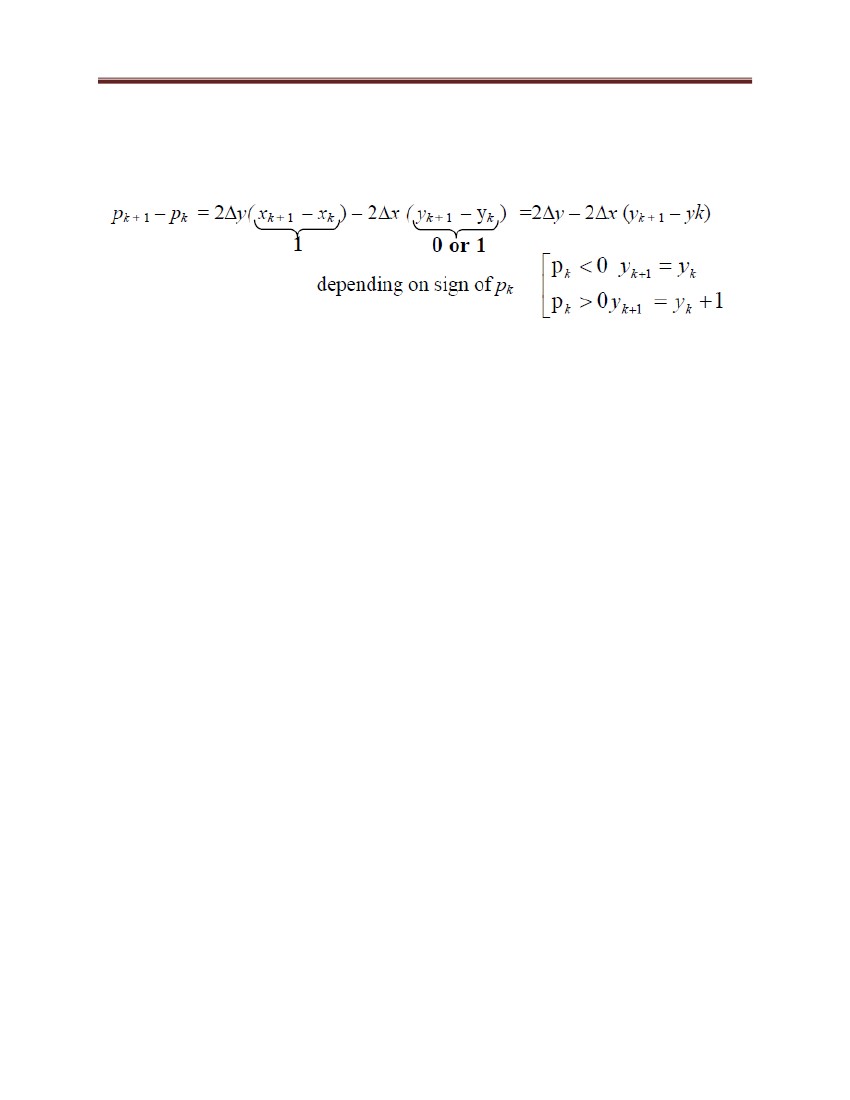
(e) Repeat step d Δx times.

Note:

• Bresenhams algorithm is generalised to lines with arbitrary slopes by considering the

symmetry between the various octants and quadrants of the coordinate system.

60



Computer Graphics and Vision

• For line with +ve slope (m) such that m > 1, then we interchange the roles of x and y

direction i.e., we step along y directions in unit steps and calculate successive x values

nearest the line path.

• for –ve slopes the procedures are similar except that now, one coordinate decreases as

the other increases.

Example 2: Draw line segment joining (20, 10) and (25, 14) using Bresenham line

generation algorithm.

Solution: (x0, y0) → (20, 10) ; (x1, y1) → (25, 14)

m = (14-10)/(25-20)=4/5==−− < 1

As, m = yΔ/xΔ = 4/5 ⇒ Δy = 4

Δx = 5

→ plot point (20, 10)

pi = 2Δy – Δx

i = 1: pi = 2 \* 4 – 5 = 3

as p1 > 0 so x0 ← 21; y0 ← 11

now plot (21,11)

i = 2 as p1 > 0

∴p2 = p1 + 2(Δy – Δx)

= 3 + 2 (4 – 5) = 3 – 2 = 1

p2 > 0 so x0 ← 22; y0 ← 12

plot (22,12)

61



Computer Graphics and Vision

i = 3 as p2 > 0

∴p3 = p2 + 2 (Δy – Δx) = 1 + 2 (4 – 5) = – 1

p3 < 0 ∴x0 ← 23

y0 ← 12

plot (23, 12)

i = 4 as p3 < 0

∴p4 = p3 + 2Δy

=–1+2\*4=7

∴ x0 ← 24; y0 ← 13

plot (24, 13)

i = 5 as p4 > 0

∴p5 = p4 + 2 (Δy – Δx)

= 7 + 2 (4 – 5) = 5

x0 ← 25; y0 ← 14

plot (25, 14)

{for i = 6, x0 will be > xi so algorithm terminates

Example 3: Illustrate the Bresenham line generation algorithm by digitizing the line with

end points (20, 10) and (30,18)

62



Computer Graphics and Vision

plot initial point (x0, y0) = (20, 10) in frame buffer now determine successive pixel positions

along line path from decision parameters value (20, 10).

Code for Bresenham line generation algorithm using C

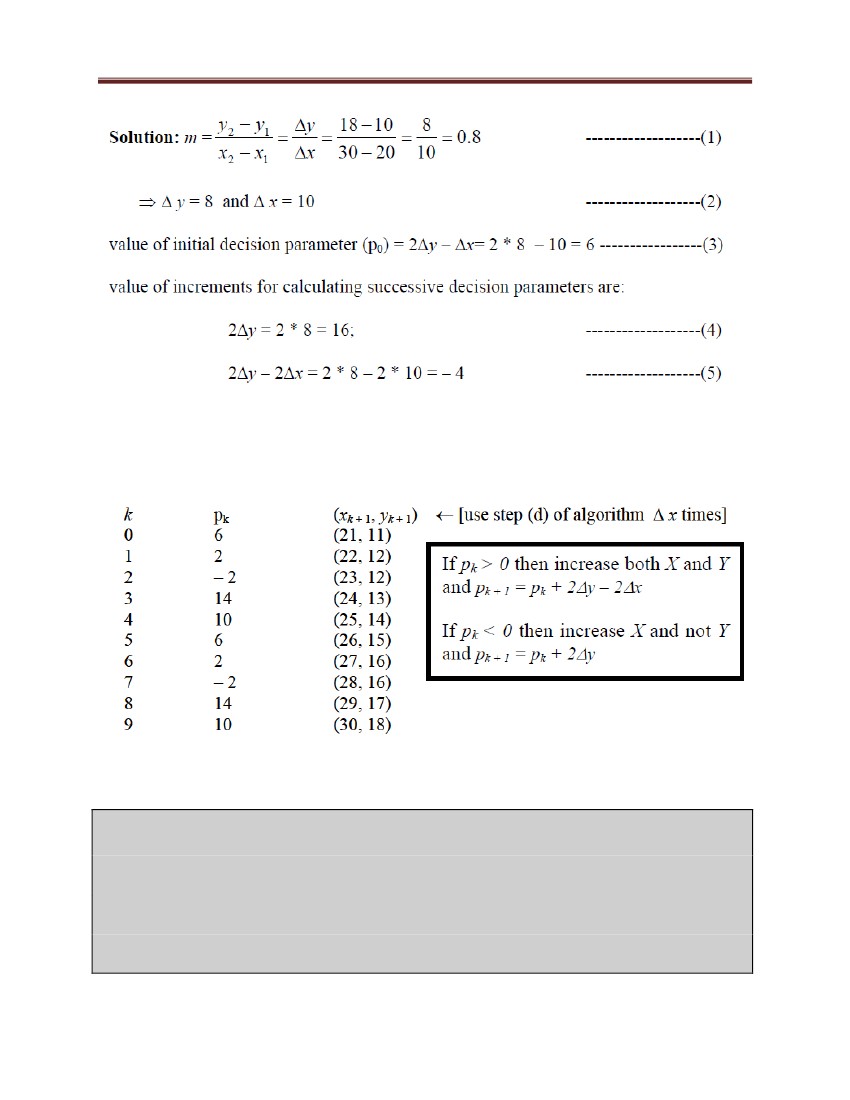
void draw\_line(int x0, int y0, int x1, int y1, int color)

{

int x, y;

int dx, dy;

63



Computer Graphics and Vision

int d;

dx = x1 - x0;

dy = y1 - y0;

d = dx - 2 \* dy;

y = y0;

for (x = x0; x <= x1; x++)

{

putpixel(x, y, color);

if (d < 0)

{

y = y + 1;

d = d + 2\*dx - 2\*dy;

}

else

{

d = d - 2\*dy;

}

}

}

4.4 Circle-generation algorithms

Circle is one of the basic graphic component, so in order to understand its generation, let us

go through its properties first:

64

Computer Graphics and Vision

4.4.1 Properties of Circles

In spite of using the Bresenham circle generation (i.e., incremental approval on basis of decision

parameters) we could use basic properties of circle for its generation.

These ways are discussed below

Figure 37: Property of circle

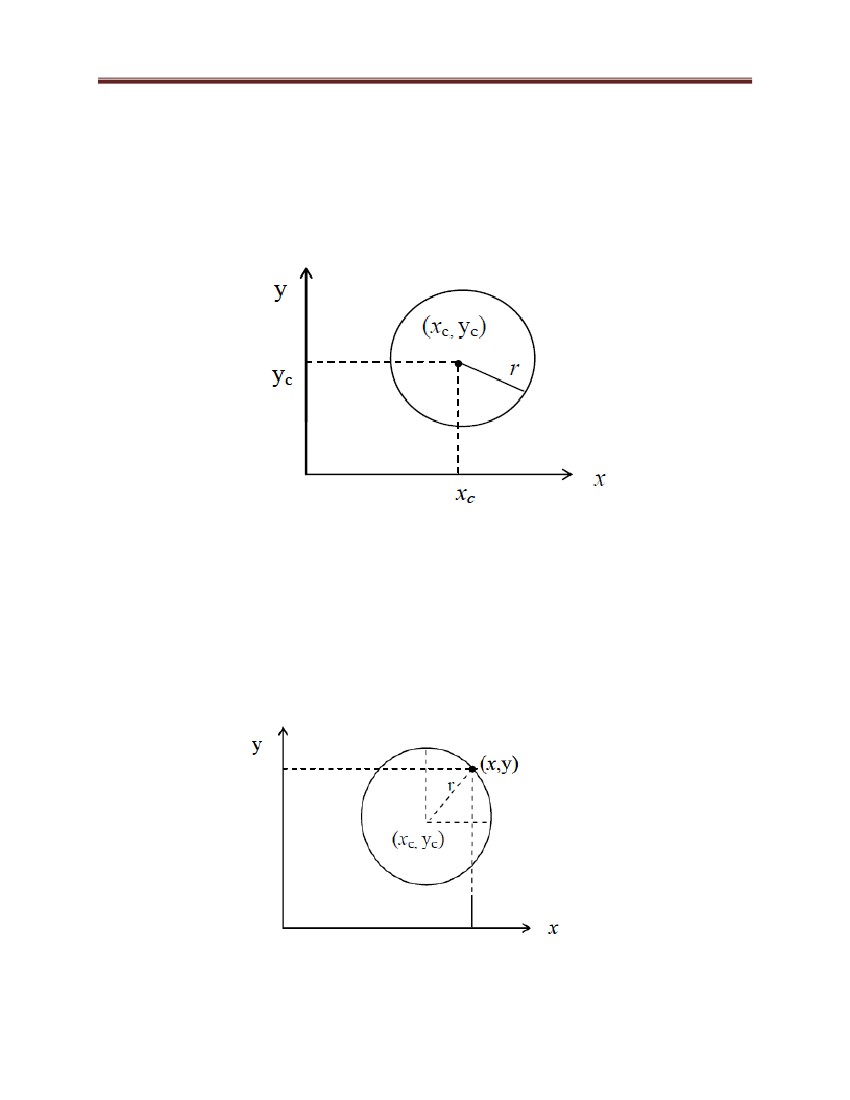
Generation on the basis of Cartesian Coordinates:

A circle is defined as the set of points or locus of all the points, which exist at a given

distance r from center (xc, yc). The distance relationship could be expressed by using

Pythagonous theorem in Cartesian coordinates as

65



Computer Graphics and Vision

Figure 38: Circle generation (cartesian coordinate system)

(x – xc) + (y – yc) = r ----------------------------------------------------------(1)

Now, using (1) we can calculate points on the circumference by stepping along x-axis

from xc–r to xc + r and calculating respective y values for each position.

2

2

2

Generation on basis of polar coordinates (r and θ)

Figure 39: Circle generation (polar coordinate system)

Expressing the circle equation in parametric polar form yields the following equations

x = xc + r cos θ

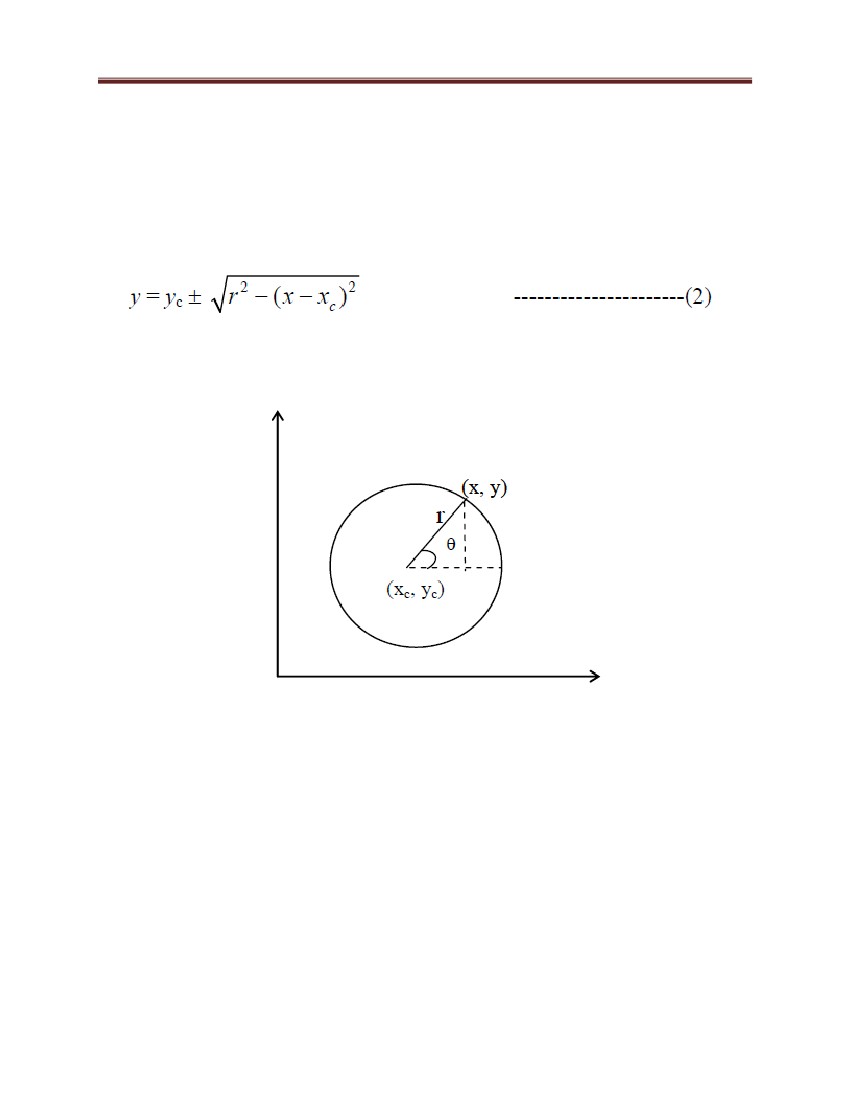
y = yc + r sin θ ----------------------------------------------------------------------------------------- (3)

Using a fixed angular step size we can plot a circle with equally spaced points along the

circumference.

Note:

66



Computer Graphics and Vision

• Generation of circle with the help of the two ways mentioned is not a good choice both require

a lot of computation equation (2) requires square root and multiplication calculations where

as equation (3) requires trigonometric and multiplication calculations. A more efficient

approach is based on incremental calculations of decision parameter. One such approach is

Bresenham circle generation. (midpoint circle algorithm).

• This Bresenham circle generation (midpoint circle algorithm) is similar to line generation.

Sample pixels at Unit x intervals and determine the closest pixel to the specified circle path at

each step.

• For a given radius and center position (x, y,) we first setup our algorithm to calculate pixel

position around the path of circle centered at coordinate origin (0, 0) i.e., we translate (xc, yc)

→ (0, 0) and after the generation we do inverse translation (0, 0) → (x c, yc) hence each

calculated position (x, y) of circumference is moved to its proper screen position by adding

xc to x and yc to y.

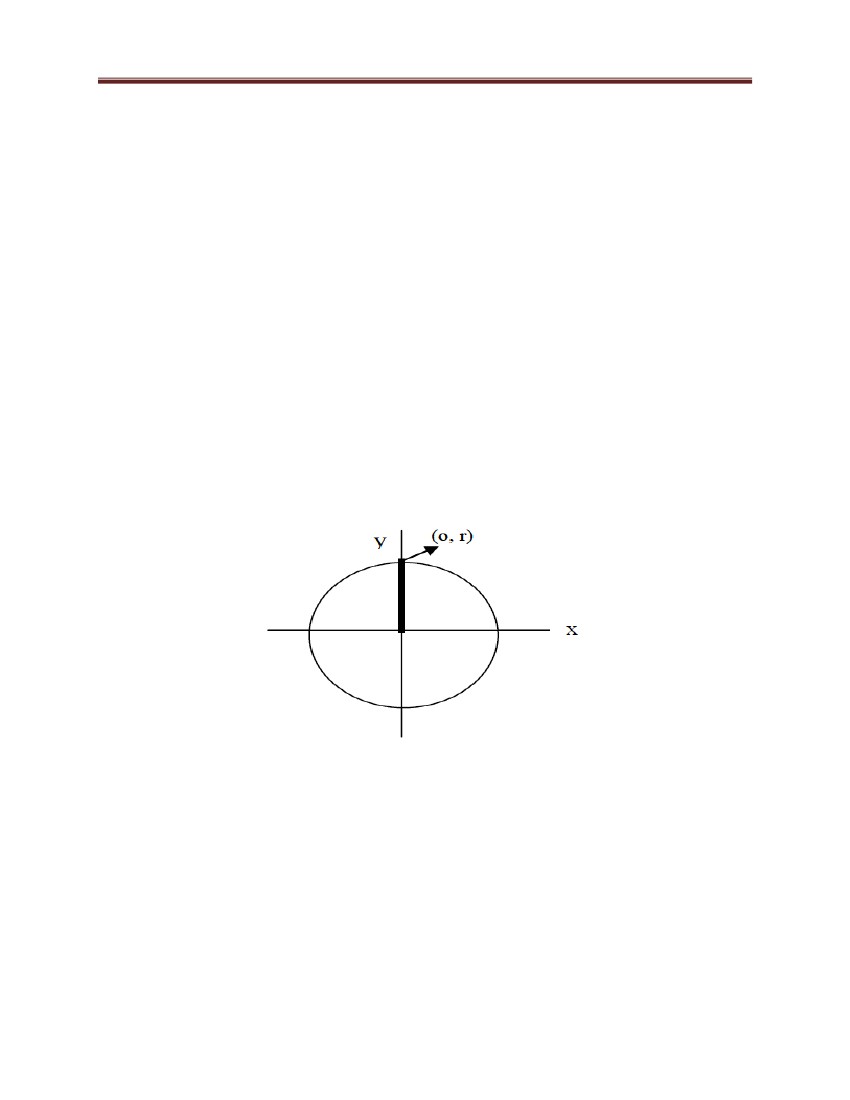
Figure 40: Circle generation (initialization)

In Bresenham circle generation (midpoint circle algorithm) we calculate points in an

octant/quadrant, and then by using symmetry we find other respective points on the

circumference

67



Computer Graphics and Vision

Figure 41: Circle generation approaches

4.4.1 Mid Point Circle Generation Algorithm

2

2

2

2

2

2

We know that equation of circle is x + y = r . By rearranging this equation, we can have

the function, to be considered for generation of circle as fc (x, y). fc (x, y) = x + y – r ---

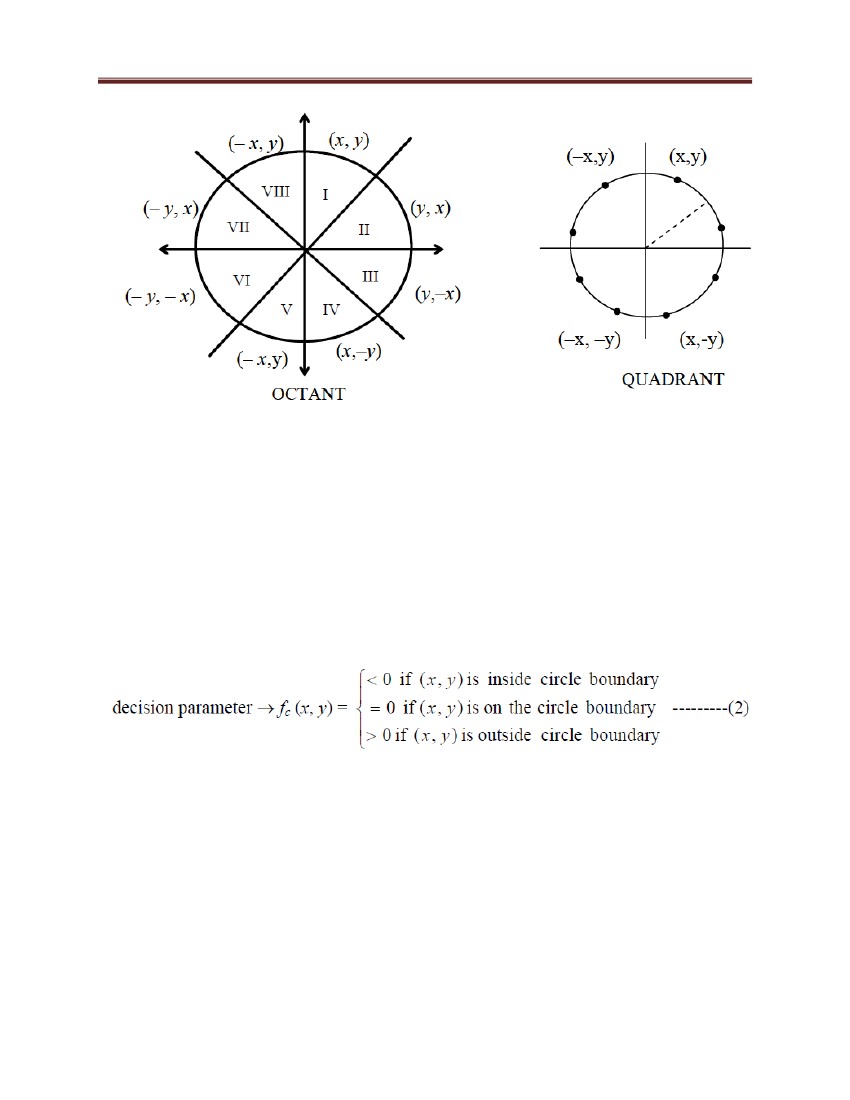
------------------------------------------------------------------------------------------------------- (1)

The position of any point (x, y) can be analyzed by using the sign of fc (x, y) i.e.,

i.e., we need to test the sign of fc (x, y) are performed for mid point between pixels near

the circle path at each sampling step

68



Computer Graphics and Vision

Figure 42: Mid-point between Candidate Pixel at sampling Position xk + 1 along Circular

Path

Figure 42 shows the mid-point of two candidate pixel (xk + 1, yk) and (xk +1, yk – 1) at

sampling position xk + 1.

Assuming we have just plotted the (xk, yk) pixel, now, we need to determine the next

pixel to be plotted at (xk + 1, yk) or (xk +1, yk – 1). In order to decide the pixel on the circles

path, we would need to evaluate our decision parameter pk at mid point between these

two pixels i.e.,

pk = fc (xk + 1, yk –1/2) = (xk + 1) + (yk – 1/2) – r ----------------------------------------------(3)

If pk < 0 then ⇒ midpoint lies inside the circle and pixel on scan line yk is closer to circle

boundary else mid point is outside or on the circle boundary and we select pixel on scan

line yk – 1 successive decision parameters are obtained by using incremental calculations.

The recursive way of deciding parameters by evaluating the circle function at sampling

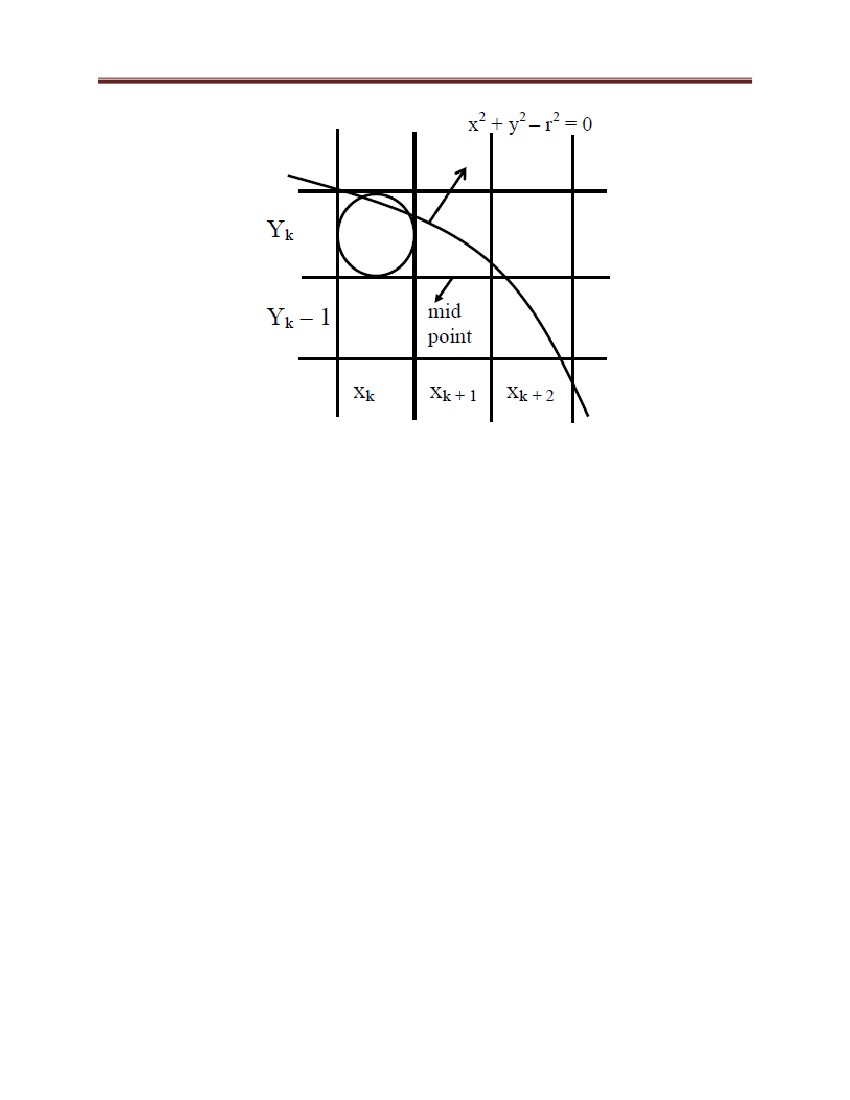
positions xk + 1 + 1 =( xk +1)+ 1= xk +2

69

2

2

2



Computer Graphics and Vision

pk + 1 = fc (xk +1 + 1, yk + 1 – 1/2) = [ (xk +1) + 1) + (yk + 1 – 1/2) – r ] --------(4)

subtract equation (3) from (4) we get

2

2

2

Here, yk + 1 could be yk or yk – 1 depending on sign of pk

How do these increments occurs?

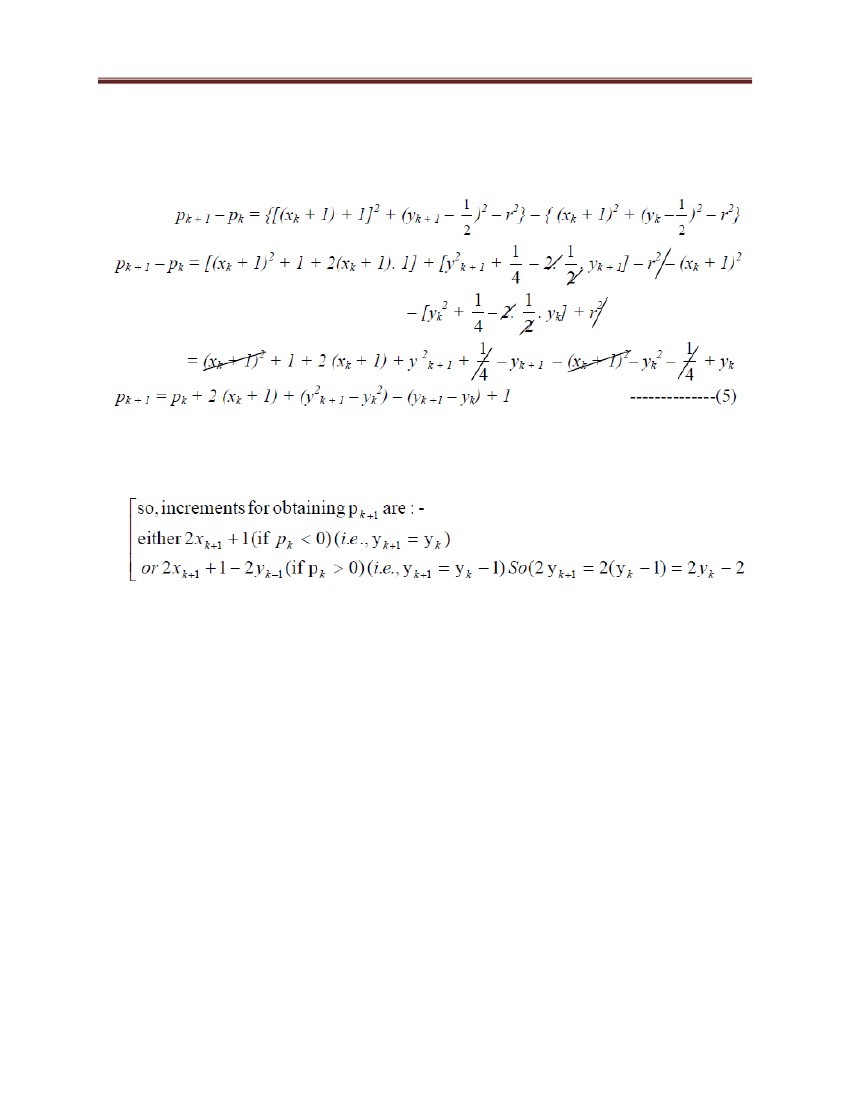
• yk + 1 = yk : use it in (5) we get

pk + 1 = pk + 2 (xk +1) + 0 – 0 + 1 = pk + (2xk + 1 + 1)

pk + 1 = pk + (2xk + 1 + 1)

………………………………………………….6

70



Computer Graphics and Vision

Note:

• At starting position (0, r) the terms in (8) and (9) have value 0 and 2r respectively. Each

successive value is obtained by adding 2 to the previous value of 2x and subtracting 2

from the previous value of 2y.

The initial decision parameter is obtained by evaluating the circle function at start

position

(x0, y0) = (0, r)

If radius r is specified as an integer then we can round p0 to

p0 = 1 – r

4.4.2 Midpoint Circle Algorithm

(a) Input radius r and circle, center (xc, yc) and obtain the first point on the circumference

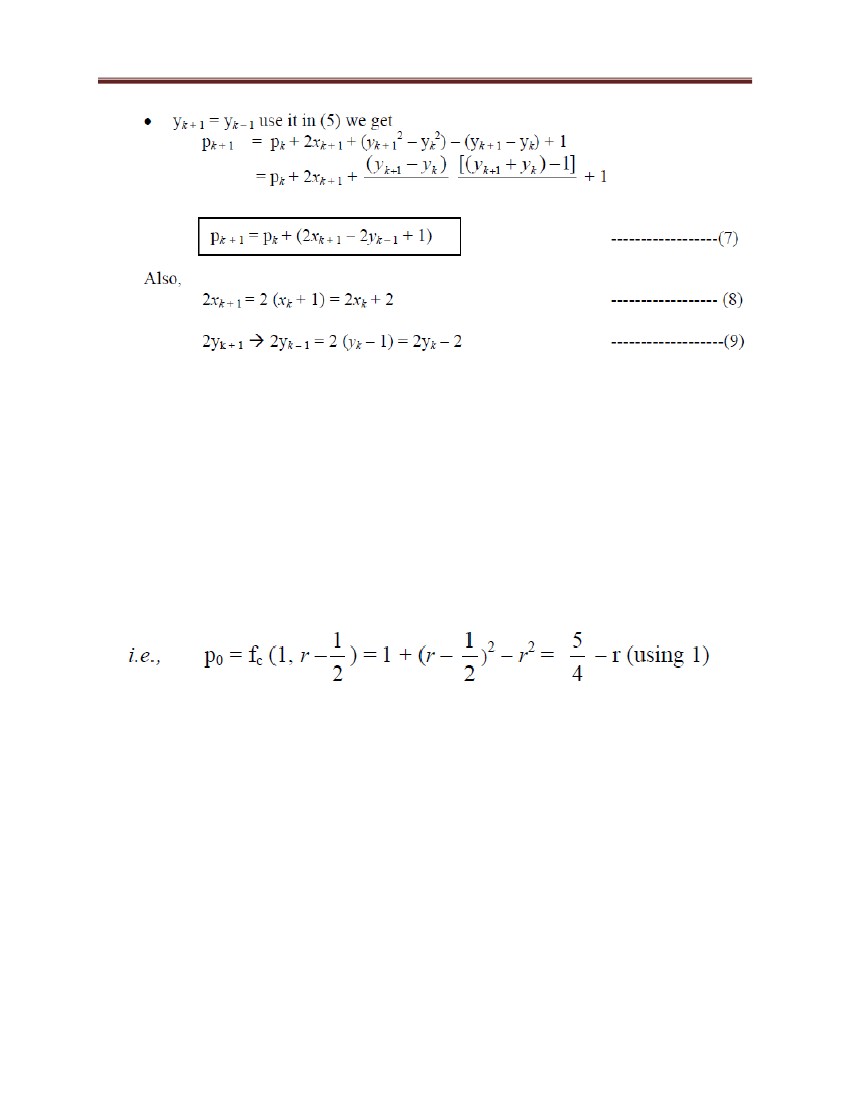
of the circle centered on origin as

(x0, y0) = (0, r)

(b) Calculate initial value of decision parameter as

71

(for r as an integer).



Computer Graphics and Vision

p0 = 4/5– r ~ 1 – r

(c) At each xk position starting at k = 0 perform following test.

1) If pk < 0 then next point along the circle centered on (0, 0) is (xk + 1, yk)

and pk + 1 = pk + 2xk + 1 + 1

2) Else the next point along circle is (xk + 1, yk – 1) and

pk + 1 = pk + 2xk + 1 – 2yk – 1

where 2xk + 1 =2(xk +1) = 2xk + 2 and 2yk – 1= 2(yk – 1) =2yk – 2

(d) Determine symmetry points in the other seven octants.

(e) Move each calculated pixel position (x, y) onto the circular path centered on (xc, yc)

and plot coordinate values x = x + xc , y = y + yc

(f) Repeat step (c) through (e) until x ≥ y.

st

Example 4: Given a circle radius r = 10 determine positions along the circle octants in 1

Quadrant from x = 0 to x = y.

Solution: An initial decision parameter p0 = 1 – r = 1 – 10 = – 9

For circle centered on coordinate origin the initial point (x0, y0) = (0, 10) and initial

increment for calculating decision parameter are:

Using mid point algorithm point are:

2x0 = 0, 2y0 = 20

72



Computer Graphics and Vision

Code for Mid Point circle algorithm using c

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

int x,y,r;

void Drawcircle(int,int,int);

printf("Enter the Mid points and Radious:");

scanf("%d%d%d",&x,&y,&r);

initgraph(&gd,&gm,"");

Drawcircle(x,y,r);

getch();

closegraph();

}

void Drawcircle(int x1,int y1,int r)

{

int x=0,y=r,p=1-r;

void cliplot(int,int,int,int);

cliplot(x1,y1,x,y);

while(x<y)

73

Computer Graphics and Vision

{

x++;

if(p<0)

p+=2\*x+1;

else

{

y--;

p+=2\*(x-y)+1;

}

cliplot(x1,y1,x,y);

}

}

void cliplot(int xctr,int yctr,int x,int y)

{

putpixel(xctr +x,yctr +y,1);

putpixel(xctr -x,yctr +y,1);

putpixel(xctr +x,yctr -y,1);

putpixel(xctr -x,yctr -y,1);

putpixel(xctr +y,yctr +x,1);

putpixel(xctr -y,yctr +x,1);

putpixel(xctr +y,yctr -x,1);

putpixel(xctr -y,yctr -x,1);

getch();

}

4.5 Polygons

We know that polygon is a figure with many sides. A polygon can be represented as a group of

connected edges, forming a closed figure. The line segments which form the boundary of a

polygon are called edges or sides. The endpoints of the edges of the polygon form the vertices. A

triangle is a simple form of polygon and three vertices. The polygon may be any shape.

74



Computer Graphics and Vision

4.5.1 POLYGON FILLING ALGORITHM

In many graphics displays, it becomes necessary to distinguish between various regions

by filling them with different color or at least by different shades of gray. There are

various methods of filling a closed region with a specified color or equivalent gray scale.

The most general classification appears to be through following algorithms:

1) Scan Line polygon fill algorithm.

2) Seed fill algorithm./Flood fill algorithm which are further classified as:

(a) Boundary fill algorithm

(b) Interior fill algorithm

We restrict our discussion to scan line polygon fill algorithm, although we will discuss

the others in brief at the end of this section.

4.5.2 Scan Line Polygon Fill Algorithm

In this, the information for a solid body is stored in the frame buffer and using that

information all pixels i.e., of both boundary and interior region are identified and, are

hence plotted.

Here we are going to do scan conversion of solid areas where the region is bounded by

polygonal lines. Pixels are identified for the interior of the polygonal region, and are then

filled plotted with the predefined color.

Let us discuss the algorithm briefly, then we will go into details on the same.

This algorithm checks and alters the attributes (i.e., characteristics and parameters) of the

pixels along the current raster scan line. As soon as it crosses over from the outside to the

inside of a boundary of the specified polygon it starts resetting the color (or gray)

attribute. In effect filling the region along that scan line. It changes back to the original

attribute when it crosses the boundary again. Figure 14 shows variations of this basic

idea.

75



Computer Graphics and Vision

Figure 43: Concept of scan line polygon filling

So as to understand Scan Line Polygon Fill Algorithm in detail consider Figure 15:

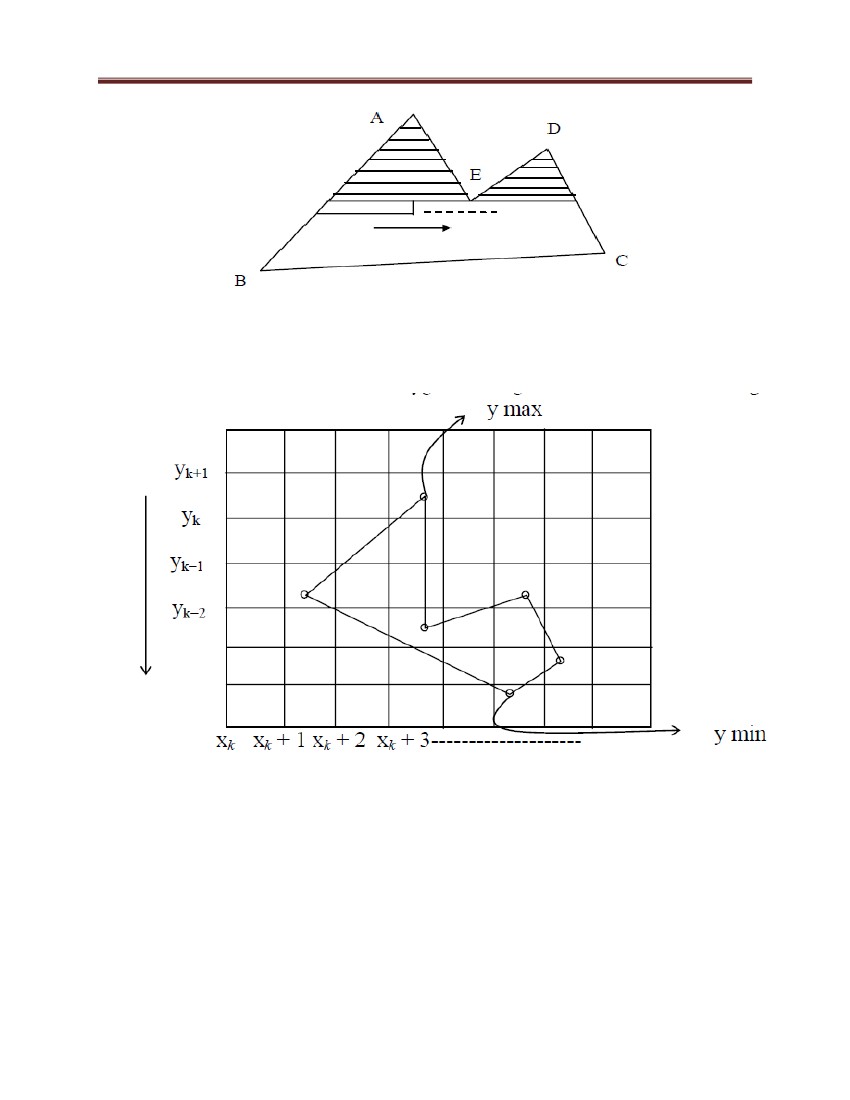
Figure 44: Scan line polygon filling

Here, in Figure 44,

• yk, yk – 1,….. → are scan lines from top to bottom (value of y at top or scan line at top

has maximum value and the scan line at bottom has minimum y value).

76



Computer Graphics and Vision

• xk, xk + 1…..→are consecutive pixels on a scan line i.e., (xk,yk), (xk+1,yk ),…………⇒ a

sea of pixels for scan lines yk, yk – 1, yk – 2, ….., yi and for the chosen scan line we find

pixels satisfying the condition as mentioned above.

• For a given scan line, let us assume that it intersects edges of the given polygonal region

at the points P1, P2, …, Pk. Sort these points P1, P2, …, Pk in terms of the X-

coordinates in increasing order.

The basic scan-line algorithm is as follows:

– Find the intersections of the scan line with all edges of the polygon

– Sort the intersections by increasing x coordinate

– Fill in all pixels between pairs of intersections that lie interior to the polygon

Now, for a chosen scan line there are three cases:

(a) Scan line passes through the edges in between shown by point a in Figure 16.

(b) Scan line passes through the vertex of the polygon whose neighbouring vertices lie

completely on one side of the scan line (point b, point c in Figure 16).

(c) Scan line passes through some bottom vertex of polygon whose neighbouring vertices

lie on both sides of the scan line

Figure 45: Cases for scan line polygon filling

77



Computer Graphics and Vision

In case a, i.e., when a scan line intersects an edge at a point but not a vertex point of

the edge (as in case of point 1, 2, 3, 4) then that point of intersection is considered as a

single point and it is taken as ON/OFF point depending on the requirement.

Therefore, in this case the intersection points are taken as 1234. Color of the pixels

between the intersection points 1 and 2 is replaced with the filling color but the color

of the pixels between the points 2 and 3 is left unchanged. Again the color of the

pixels between 3 and 4 are changed by the filling color.

In case b and c, case c i.e., when a scan line intersects an edge E1 at a vertex V1 i.e.,

a vertex point of the edge E1 whose y coordinate is greater (or less ) than the y

coordinate values of the other two vertex points V2 and V3 where, for example, edge

E1 has end vertex points V1, V2 and edge E2 having end vertex points V1, V3. That

is the vertices V2 and V3 will lie either completely above V1 or both V2 and V3 will

lie completely below the vertex V1. In this case, the point of intersection, i.e., E1, will

be counted two times. For example, the vertices 1′ and 2′ are such that their y-

coordinate values are greater than the y-coordinate values of their neighboring

vertices and therefore they are counted twice.1′1′ 2′2′ i.e. the color of the pixels

between 1′, 1′ is replaced with the filling color but the color of the pixels between 1′ ,

2′ is left unchanged. Again the color of the pixels between 2′, 2′ is changed.

Assume that the scan line passes through a vertex point V1 of the polygonal region

having neighboring vertices V0 and V2., i.e. let E1 and E2 be two edges of the

polygon so that V0, V1 and V1, V2 be respectively their end vertices. Suppose we

assume that the vertex V1 is such that the y-coordinate for the neighboring vertex V0

is grater than the y-coordinate for V1 but the y-coordinate for the neighboring vertex

V2 is less than the y-coordinate for V1. In this case, the intersection vertex point V0

will be taken as a single point and the algorithm will remain the same as above.

78



Computer Graphics and Vision

The point of intersection is to be considered two times, i.e., 1′′ 2′′ 2′′ 3′′. 1′′ 2′′ ⇒

make pixels ON from 1′′ to 2′′, 2′′ 2′′ don‘t make pixel ON, 2′′, 3′′⇒ make pixels ON

from 2′′ to 3′′.

Now, the requirements for scanning an image are:

1) Determine the point of intersection of an edge and scan line

2) Sort the points on scan line w.r.t x value. Get the respective pixels ON

Figure 46: Requirement for image scanning

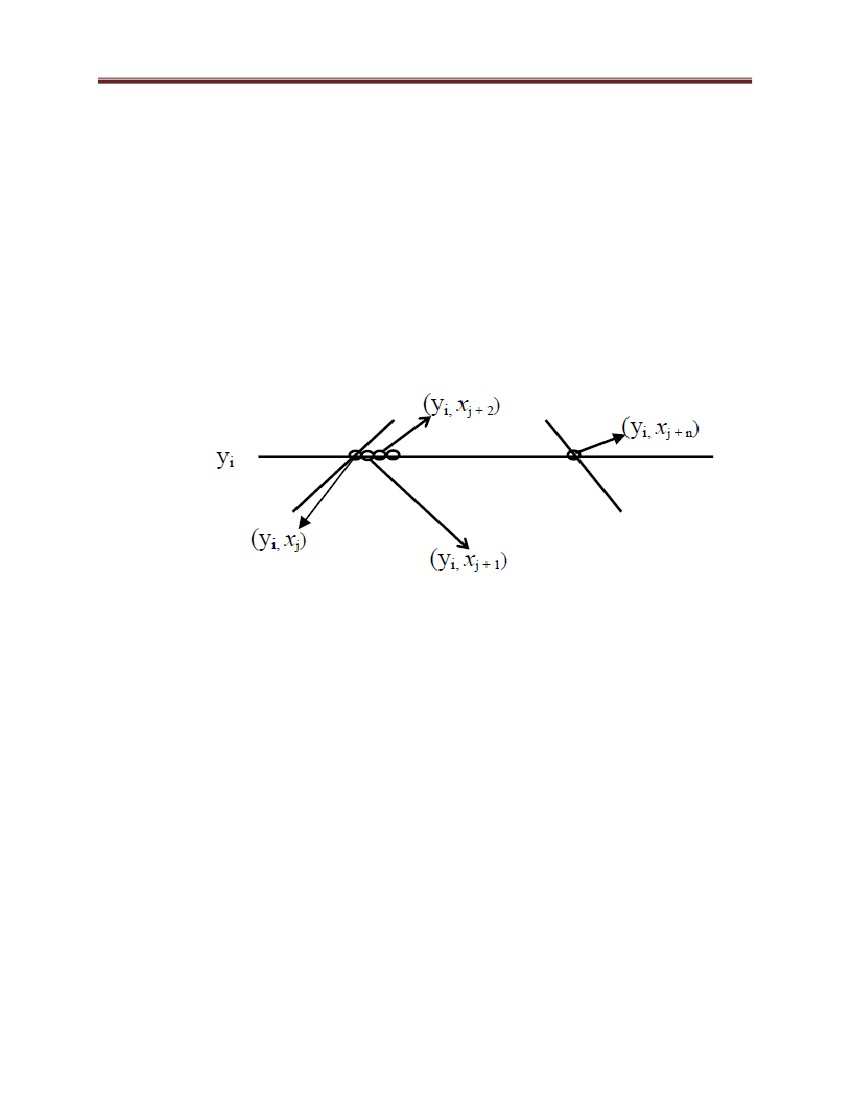
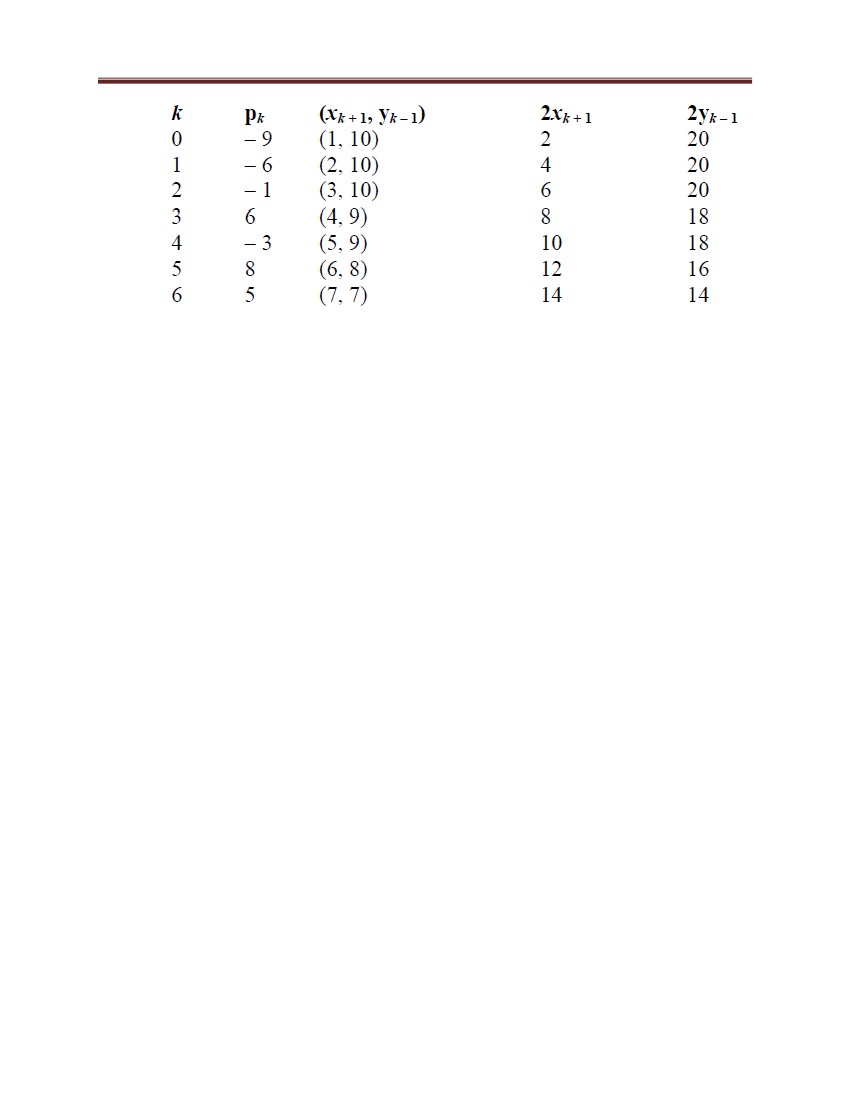
Sum up: To fill a polygonal image we have to identify all edges and vertices of

intersections while moving from scan line ymax to ymin, for each scan line. Then check

for the case and perform the algorithm accordingly.

Recursive way to scan a figure

79



Computer Graphics and Vision

Figure 47: Scan line polygon filling (a recursive approach)

For a scan line y = yi identify the edges and use the data of the point of intersection (xi

,yi.). Say x0 is point on scan line y0 and x1 is point on scan line y1. Now between (x0,

y0) and (x1, y1) we find slope (this all is to find recursive way to scan a figure).

5 Transformations

Transformations are a fundamental part of computer graphics. The transformation

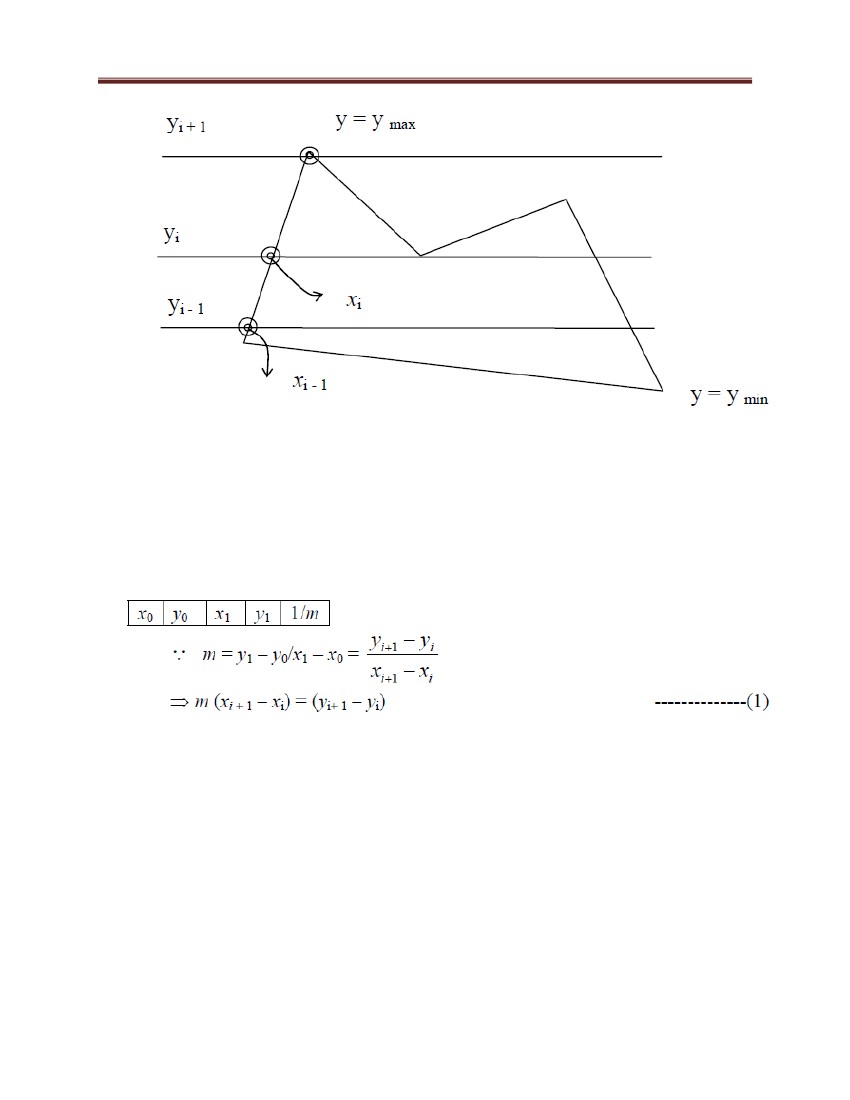
process refers to the linear transformation from 3D space to a 2D viewing space.

Transformations are used to position objects, to shape objects, to change viewing

positions, and even to change how something is viewed. Ultimately, each vertex position

must be transformed from its defining object space to the device coordinates (pixel

80



Computer Graphics and Vision

space). This often involves a combination of rotations, translations, scales, and

perspective transformations.

5.1 2D Transformations

A point p in 2D is represented as a pair of numbers: p= (x, y) where x is the x-coordinate

of the point p and y is the y-coordinate of p . 2D objects are often represented as a set of

points (vertices), {p1,p2,...,pn}, and an associated set of edges {e1,e2,...,em}. An edge is

defined as a pair of points e = {pi,pj}. What are the points and edges of the triangle

below?

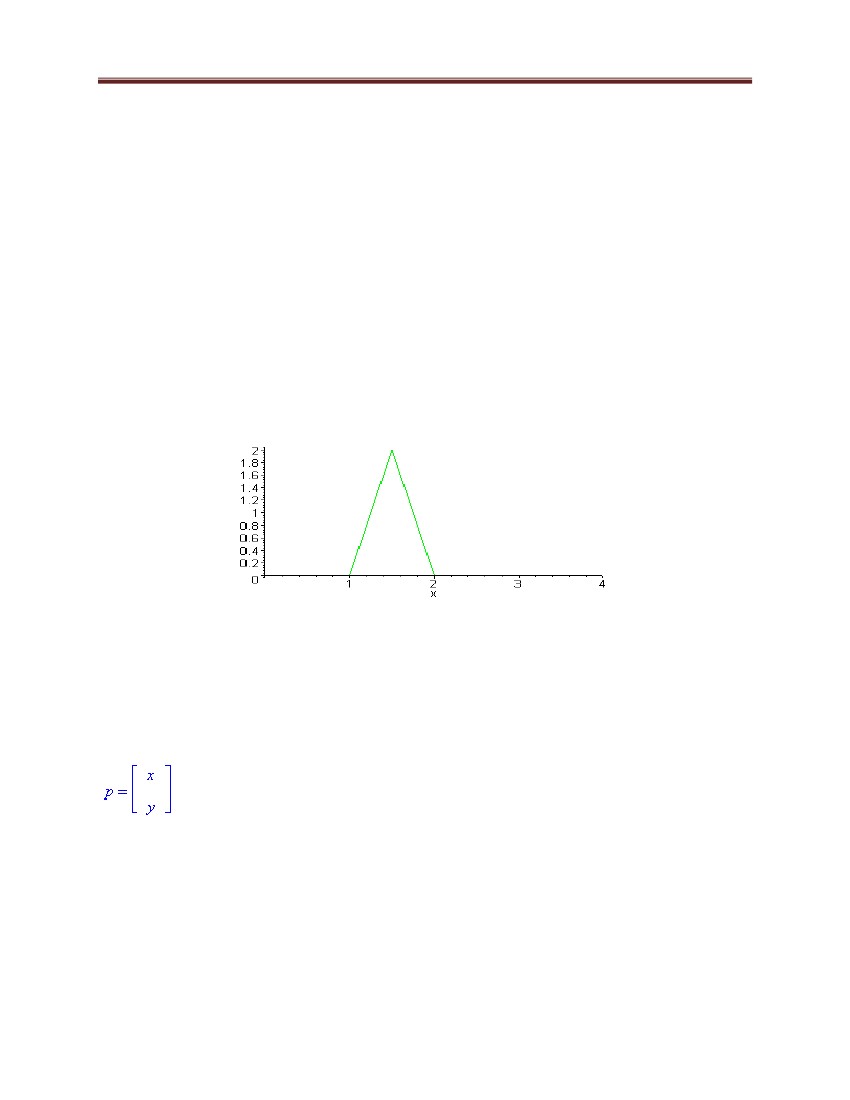
Figure 48: Example of the transformation

We can also write points in vector/matrix notation as

Here, we first discuss general procedures for applying translation, rotation, and scaling

parameters to reposition and resize two-dimensional objects.

81



Computer Graphics and Vision

5.2 Combining Transformation

We saw that the basic scaling and rotating transformations are always with respect to the

origin. To scale or rotate about a particular point (the fixed point) we must first translate

the object so that the fixed point is at the origin. We then perform the scaling or rotation

and then the inverse of the original translation to move the fixed point back to its original

position. For example, if we want to scale the triangle by 2 in each direction about the

point fp = (1.5,1), we first translate all the points of the triangle by T = (-1.5,1), scale by

2 (S) , and then translate back by -T=(1.5,1). Mathematically this looks like

q=

=

(

+

)+

Order Matters!

Notice the order in which these transformations are performed. The first (rightmost)

transformation is T and the last (leftmost) is -T. If you apply these transformations in a

different order then you will get very different results. For example, what happens when

you first apply T followed by -T followed by S? Here T and -T cancel each other out and

you are simply left with S

Sometimes (but be careful) order does not matter, For example, if you apply multiple 2D

rotations, order makes no difference:

R1 R2 = R 2 R1

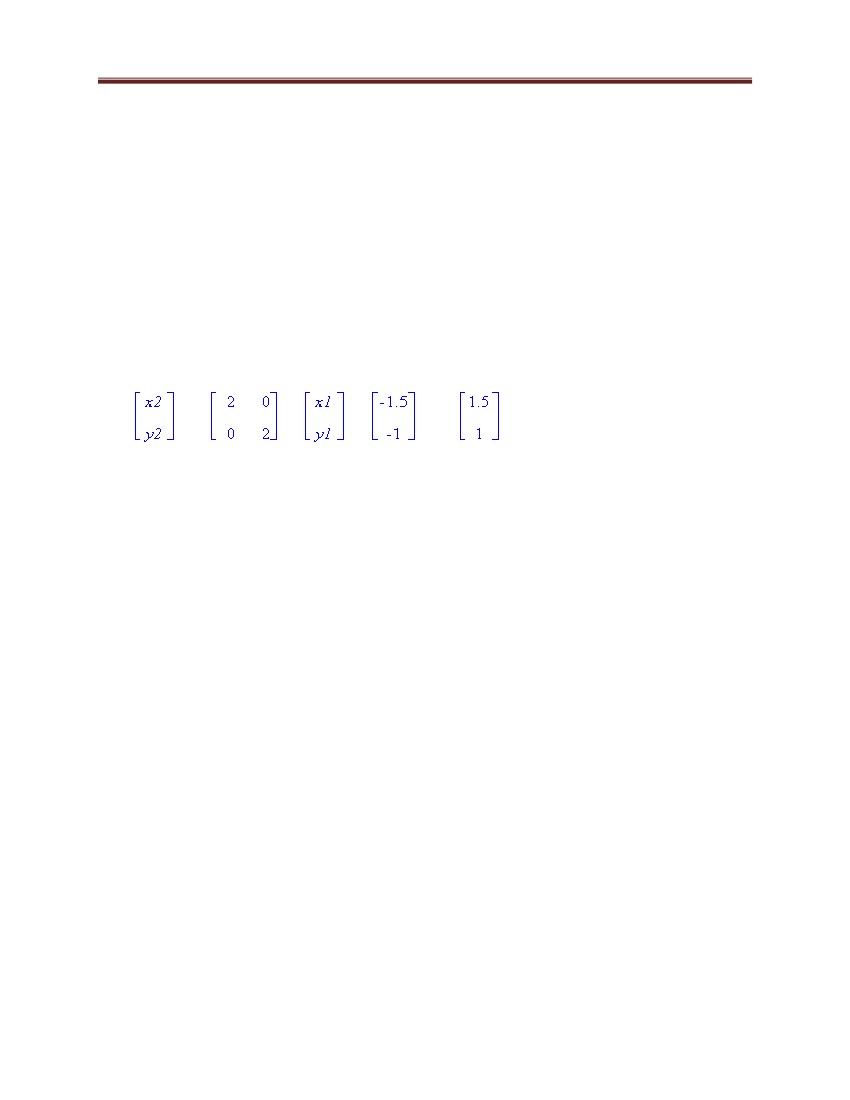
5.3 Matrix representations and homogeneous coordinates

Many graphics applications involve sequences of geometric transformations. An

animation, for example, might require an object to be translated and rotated at each

increment of the motion. In design and picture construction applications, we perform

82



Computer Graphics and Vision

translations, rotations, and scaling to fit the picture components into their proper

positions. Here we consider how the matrix representations. Matrix Representations

discussed in the previous sections can be reformulated so that such transformation

homogeneous Coordinates sequences can be efficiently processed.

Expressing positions in homogeneous coordinates allows us to represent all geometric

transformation equations as matrix multiplications. Coordinates are represented with

three-element column vectors, and transformation operations are written as 3 by 3

matrices. For translation, we have

or

q=Tp=

Exercise

=

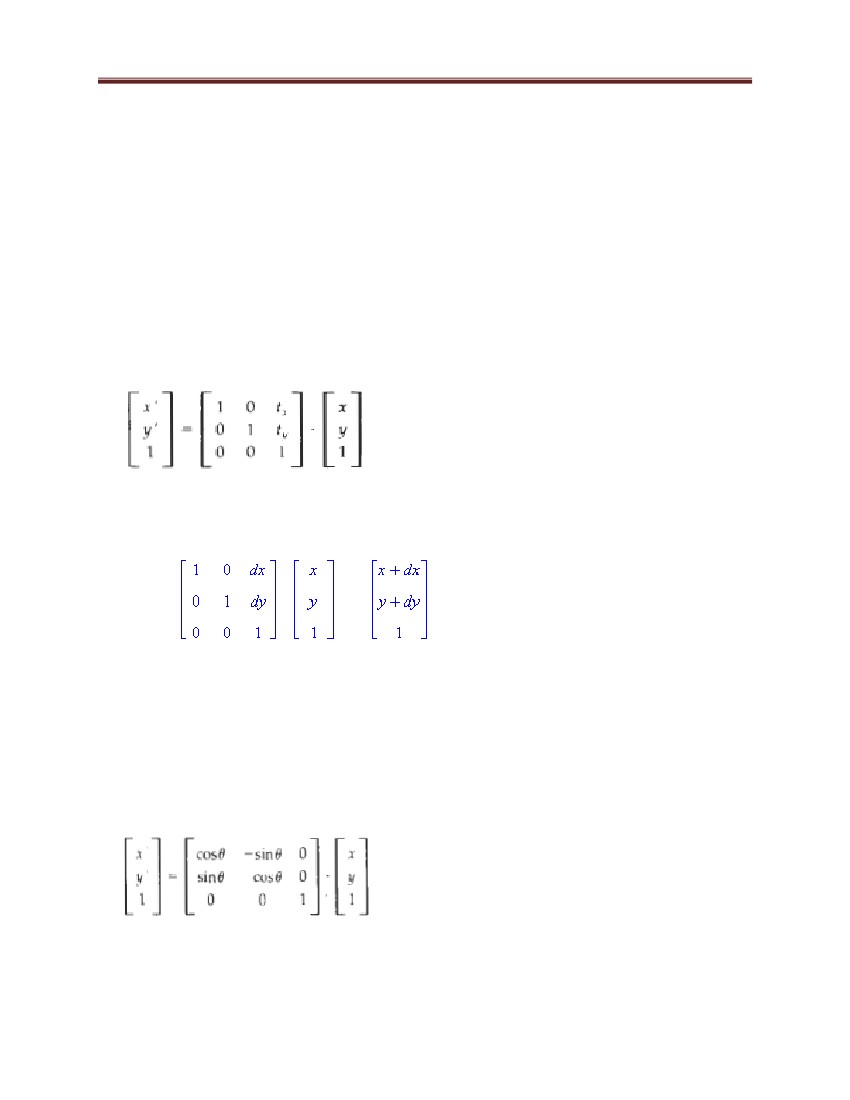
Translate a square ABCD with coordinates A(0,0), B(5,0), C(5,5),D(0,5) into

homogeneous coordinates.

Similarly, rotation transformation equations about the coordinate origin are now written

as

83



Computer Graphics and Vision

Finally, a scaling transformation relative to the coordinate origin is now expressed as the

matrix multiplication

5.4 3D Transformations

Figure: 3D Rendering Pipeline

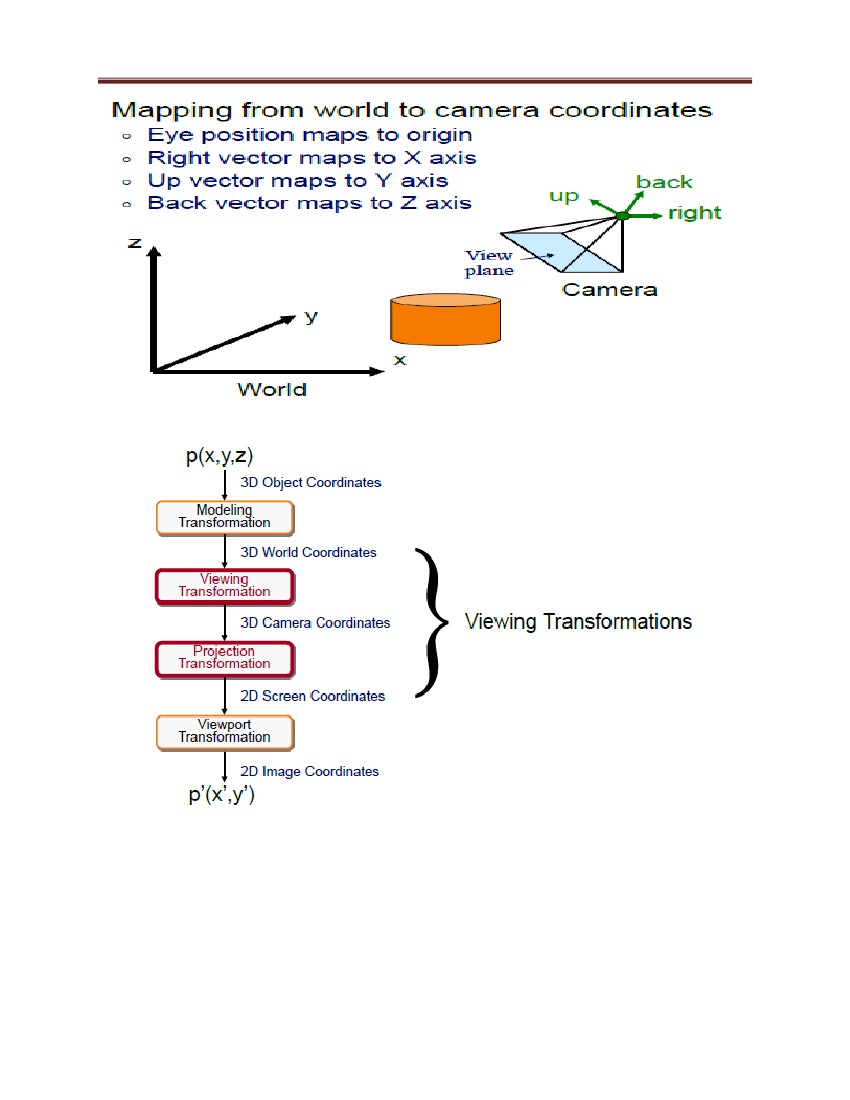
84



Computer Graphics and Vision

Figure: Viewing Transformations

85

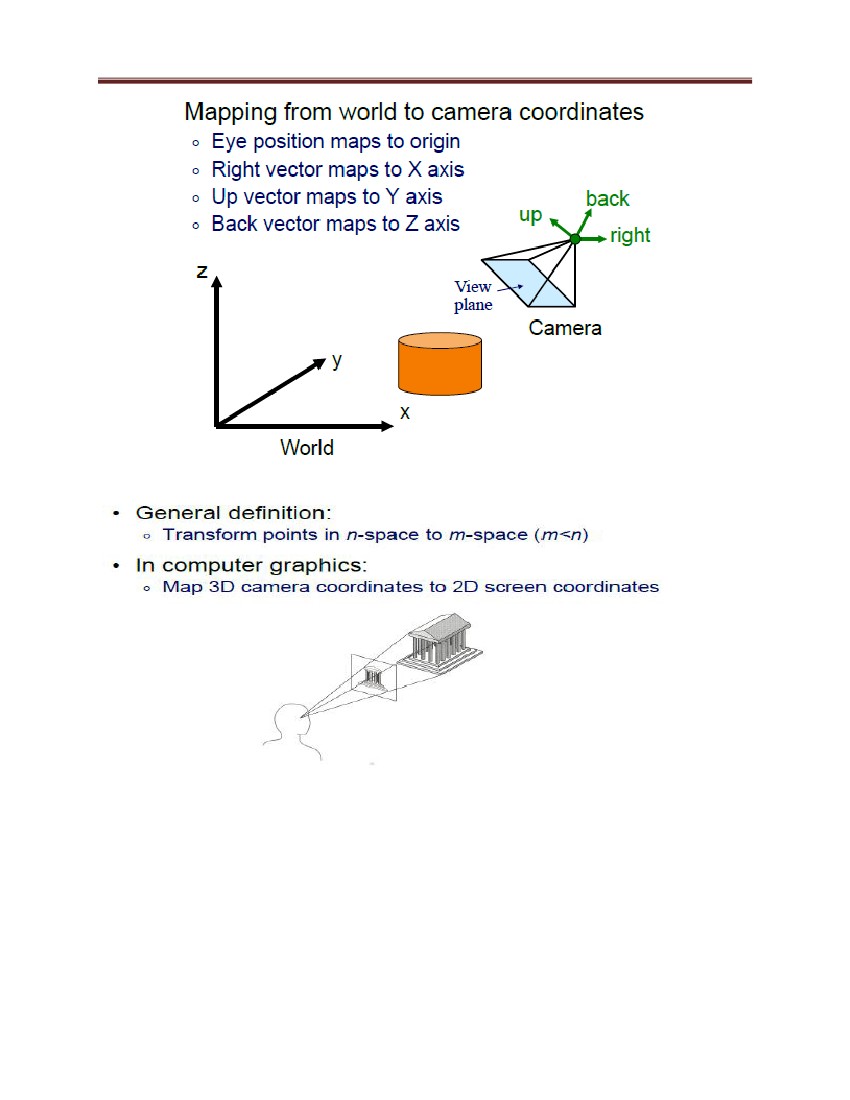


Computer Graphics and Vision

Figure: Projection

Figure: Parallel Projection

86



Computer Graphics and Vision

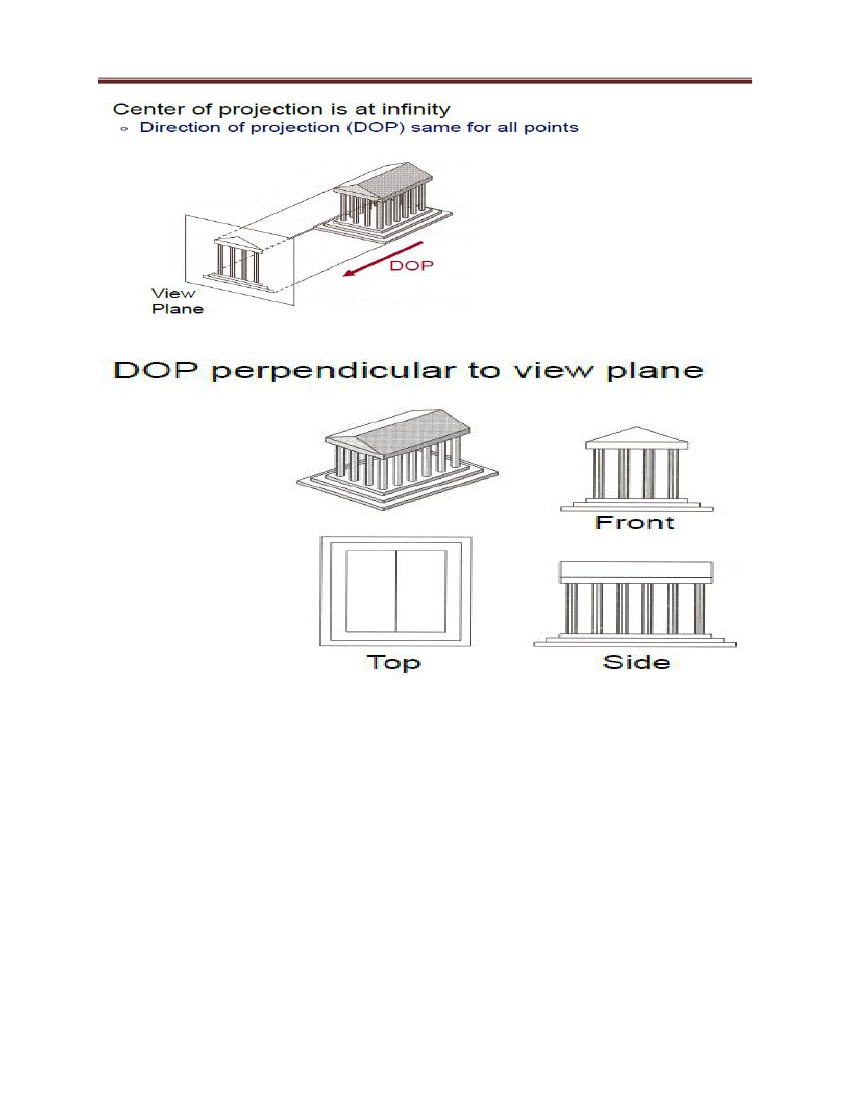
Figure: Orthographic Projections

Perspective Projection

Map points onto ―view plane‖ along ―projectors‖ emanating from ―center of projection‖

(COP)

87



Computer Graphics and Vision

5.5 Window-to-Viewport Coordinate Transformation

Once object descriptions have been transferred to the viewing reference frame, we choose

the window extents in viewing coordinates and select the viewport limits in normalized

coordinates. Object descriptions are then transferred to normalized device coordinates.

We do this using a transformation that maintains the same relative placement of objects

in normalized space as they had in viewing coordinates. If a coordinate position is at the

center of the viewing window, for instance, it will be displayed at the center of the

viewport.

Viewport:

Area on screen to be used for drawing:

Unit: pixels (screen coordinates)

Note: y-axis often points down

Window:

Virtual area to be used by application, Unit: km, mm,… (world coordinates)

Viewport Transformation

Transform 2D geometric primitives from screen coordinate system (normalized device

coordinates) to image coordinate system (pixels)

88

Computer Graphics and Vision

Figure 49: Illustration of window-to-Viewport

Figure below illustrates the window-to-viewport mapping. A point at position (my,w) in

the window 1s mapped into position (xv, yv) in the associated viewport.

From normalized coordinates, object descriptions are mapped to the various display

devices. Any number of output devices can be open in a particular application, and

another window-to-viewport transformation can be performed for each open output

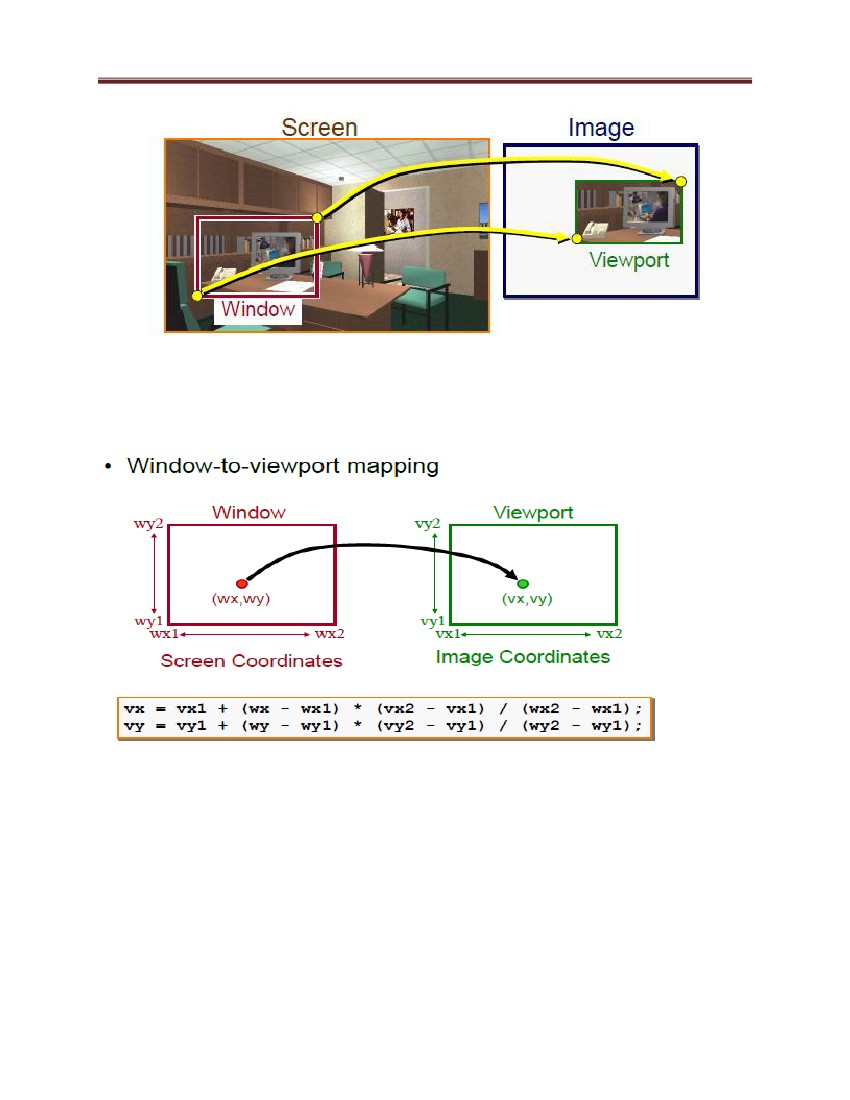
device. This mapping, called the workstation transformation, is accomplished by

selecting a window area in normalized space and a viewport area in the coordinates of the

display device. With the workstation transformation, we gain some additional control

over the positioning of parts of a scene on individual output devices.

89



Computer Graphics and Vision

Figure 50: Summary of Transformations

5.6 Clipping operations

Some triangles will be completely visible on the screen, while others may be completely

out of view. Some may intersect the side of the screen and require special handling. The

camera‘s viewable space forms a volume called the view volume. Triangles that intersect

the boundary of the view volume must be clipped.

The related process of culling refers to the determination of which primitives are

completely invisible. The output of the clipping/culling process is a set of visible

triangles that lie within the dimensions of the display device.

Generally, any procedure that identifies those portions of a picture that are either

inside or outside of a specified region of space is referred to as a clipping algorithm,

or simply clipping.

The region against which an object is to be clipped is called a clip window.

Applications of clipping include extracting part of a defined scene for viewing;

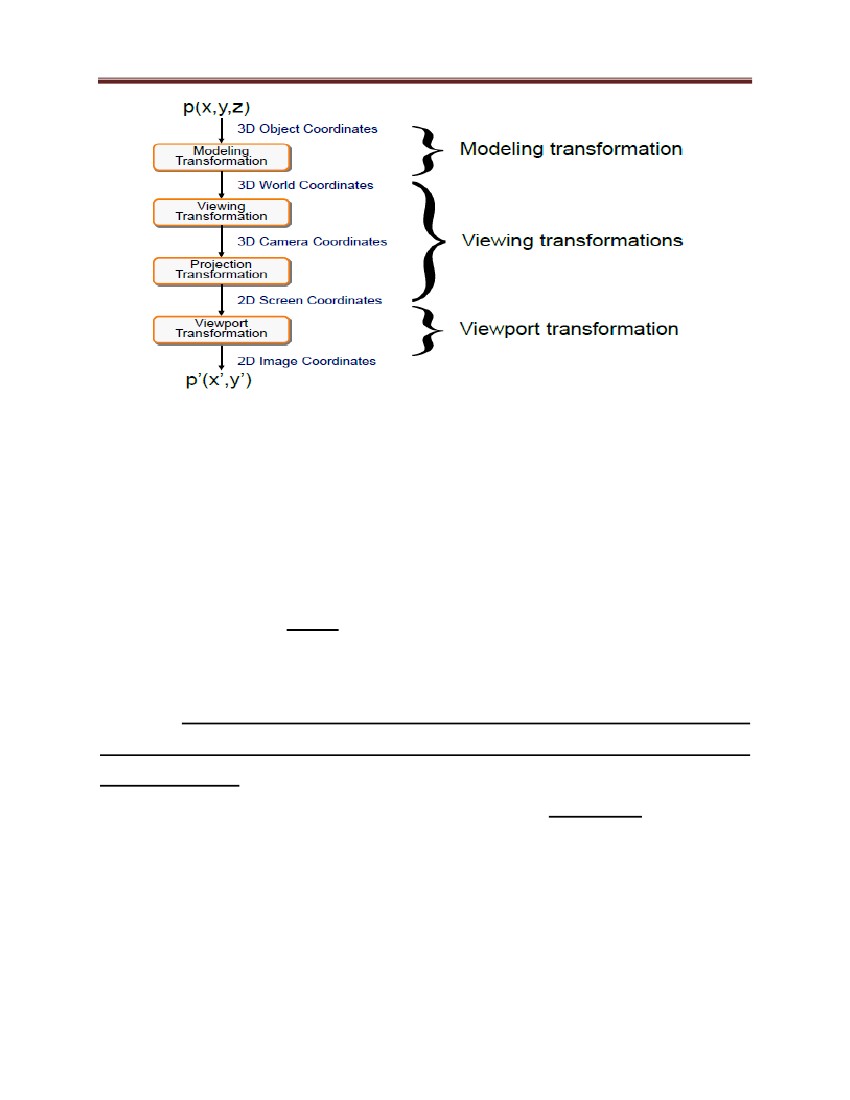
identifying visible surfaces in three-dimensional views; creating objects using solid-

modeling procedures; displaying a multi window environment; and drawing and painting

operations that allow parts of a picture to be selected for copying, moving, erasing, or

duplicating.

90



Computer Graphics and Vision

Depending on the application, the clip window can be a general polygon or it can even

have curved boundaries. We first consider clipping methods using rectangular clip

regions, and then we discuss methods for other &region shapes.

For the viewing transformation, we want to display only those picture parts that are

within the window area. Everything outside the window is discarded. Clipping algorithms

can be applied in world coordinates, so that only the contents of the window interior are

mapped to device coordinates. Alternatively, the complete world-coordinate picture can

be mapped first to device coordinates, or normalized device coordinates, then clipped

against the viewport boundaries. World-coordinate clipping removes those primitives

outside the window from further consideration, thus eliminating the processing necessary

to transform those primitives to device space. Viewport clipping, on the other hand, can

reduce calculations by allowing concatenation of viewing and geometric transformation

matrices. But viewport clipping does require that the transformation to device coordinates

be performed for all objects, including those outside the window area. On raster systems,

clipping algorithms are often combined with scan conversion.

In the following sections, line and polygon clipping routines are standard components of

graphics packages, but many packages accommodate curved objects, particularly spline

curves and conics, such as circles and ellipses. Another way to handle curved objects is to

approximate them with straight-line segments and apply the line- or polygon clipping

procedure.

5.6.1 Hidden surface removal

Drawing polygonal faces on screen consumes CPU cycles. We cannot see every surface

in scene. To save time, draw only surfaces we see.

A point is visible if there exists a direct line-of sight to it, unobstructed by another other

objects (visible surface determination).

• Moreover, some objects may be invisible because there are behind the camera, outside

of the field-of-view, too far away (clipping) or back faced (backface culling).

Hidden surfaces: why care?

91



Computer Graphics and Vision

Occlusion: Closer (opaque) objects along same viewing ray obscure more distant ones

• Reasons for removal

– Efficiency: As with clipping, avoid wasting work on invisible objects

– Correctness: The image will look wrong if we don‘t model occlusion properly

Surfaces we cannot see, and their elimination methods

-Occluded surfaces: hidden surface removal (visibility)

-Back faces: back face culling

A correct rendering requires correct visibility calculations

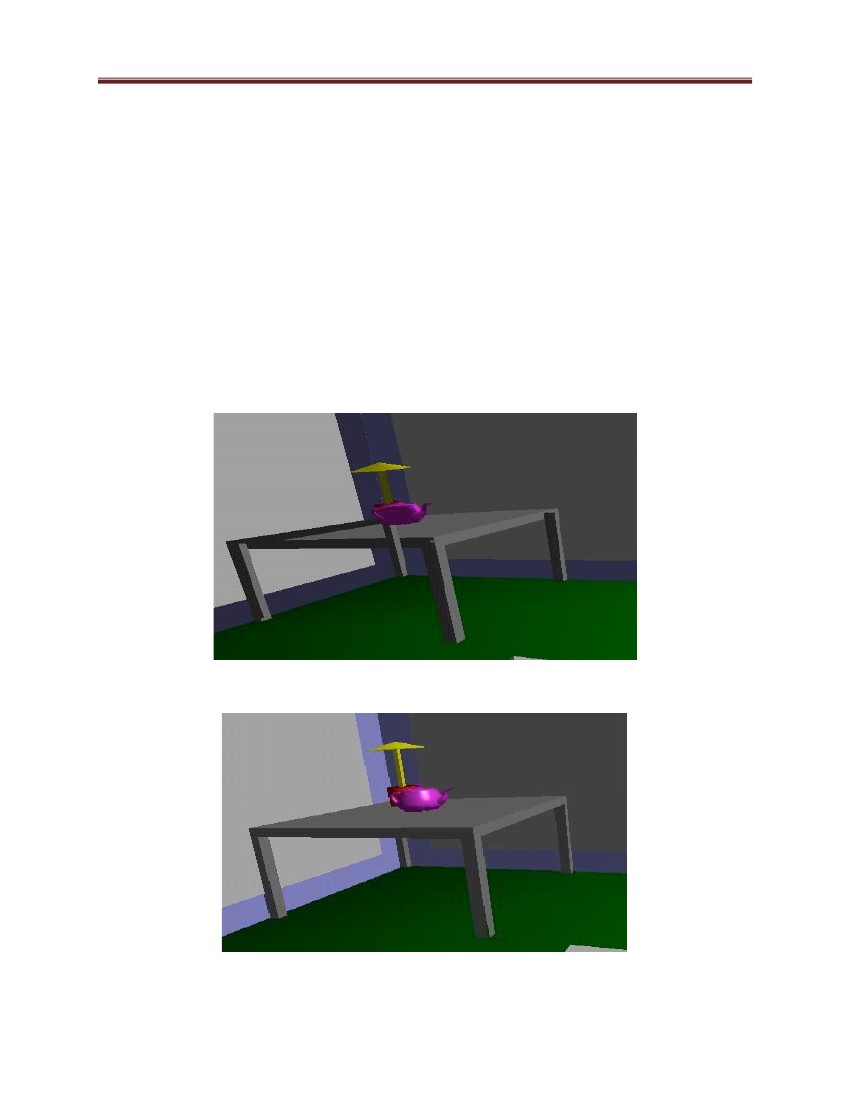
When multiple opaque polygons cover the same screen space, only the closest one is

visible (remove the other hidden surfaces)

Figure 51: Wrong visibility illustration

Figure 52: Correct visibility illustration

92



Computer Graphics and Vision

Goal

Determine which objects are visible to the eye

Determine what colors to use to paint the pixels

Active area of research

Lots of algorithms have been proposed in the past (and is still a hot topic)

Back Face Culling

Back faces: faces of an opaque object which are "pointing away" from the viewer

Back face culling: Remove back faces

Figure 53: Backface culling

If we find a back face, do not draw

Save rendering resources!

There must be other forward face(s) closer to eye

F is face of object we want to test if back face

P is a point on F

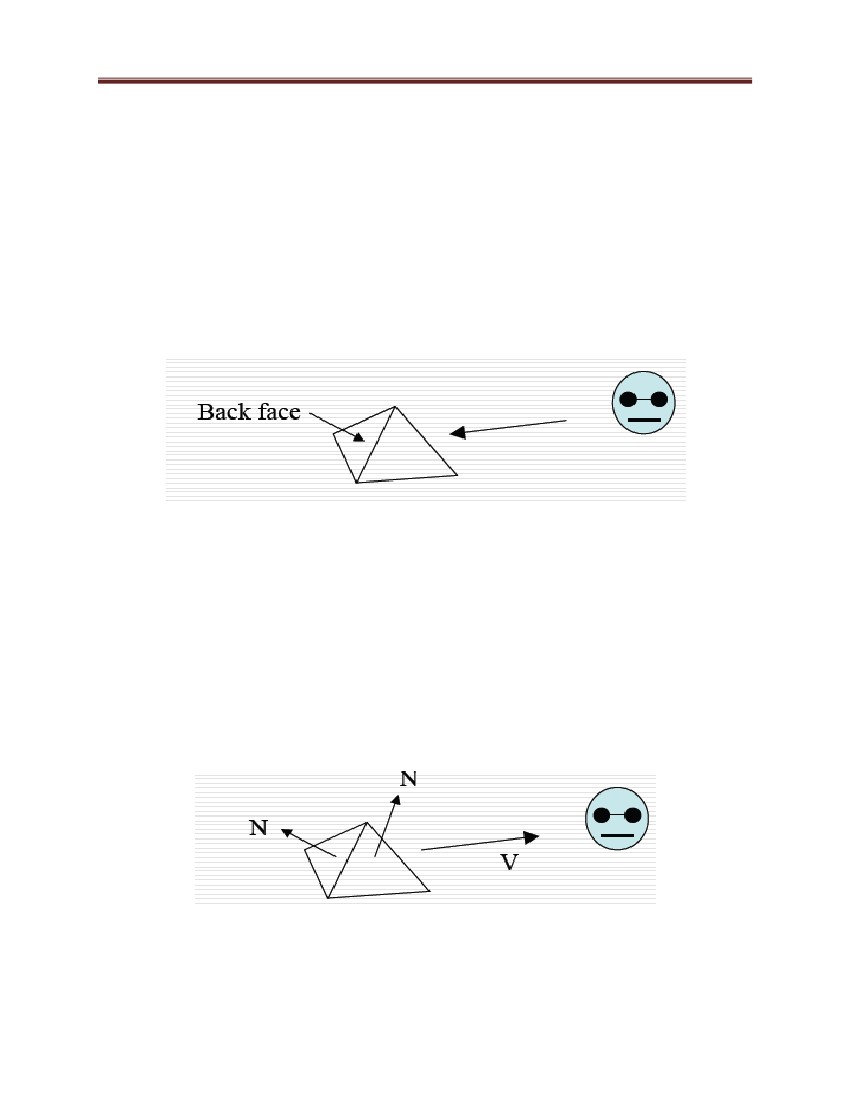
Form view vector, V as (eye – P)

N is normal to face F

Figure 53 (b): Backface culling

Back face test: F is back face if N.V < 0

93



Computer Graphics and Vision

Hidden surface removal algorithms

1) Painter‘s algorithm

2) Binary space partitioning

3) Z-buffer

4) Ray casting

a) Z-buffer

Method used in most of graphics hardware (and thus OpenGL)

Basic Z-buffer idea: add Z to frame buffer, when a pixel is drawn, check whether it is

closer than what‘s already in the frame buffer. Rasterize every input polygon;

For every pixel in the polygon interior, calculate its corresponding z value (by

interpolation). Track depth values of closest polygon (smallest z) so far. Paint the pixel

with the color of the polygon whose z value is the closest to the eye

Z-buffer approach

Z-buffer (or depth buffer) algorithm

Requires lots of memory

Recall

After projection transformation, in viewport transformation

(x,y) used to draw screen image, mapped to viewport

z component is mapped to pseudo-depth with range [0,1]

Objects/polygons are made up of vertices

Hence, we know depth z at polygon vertices

Point on object seen through pixel may be between vertices

Need to interpolate to find z

How do we choose the polygon that has the closed Z for a given pixel?

Example: eye at Z = 0, farther objects have increasingly positive values, between 0 and 1

1. Initialize (clear) every pixel in the Z buffer to 1.0

2. Track polygon Zs

94



Computer Graphics and Vision

3. As we rasterize polygons, check to see if polygon‘s Z through this pixel is less than

current minimum Z through this pixel

4. Run the following loop:

foreach polygon in scene {

foreach pixel (x,y) inside the polygon projection {

if( z\_polygon\_pixel( x, y ) < z\_buffer( x, y )) {

z\_buffer( x, y ) = z\_polygon\_pixel( x, y );

color\_buffer( x, y ) = polygon color at ( x, y )

}

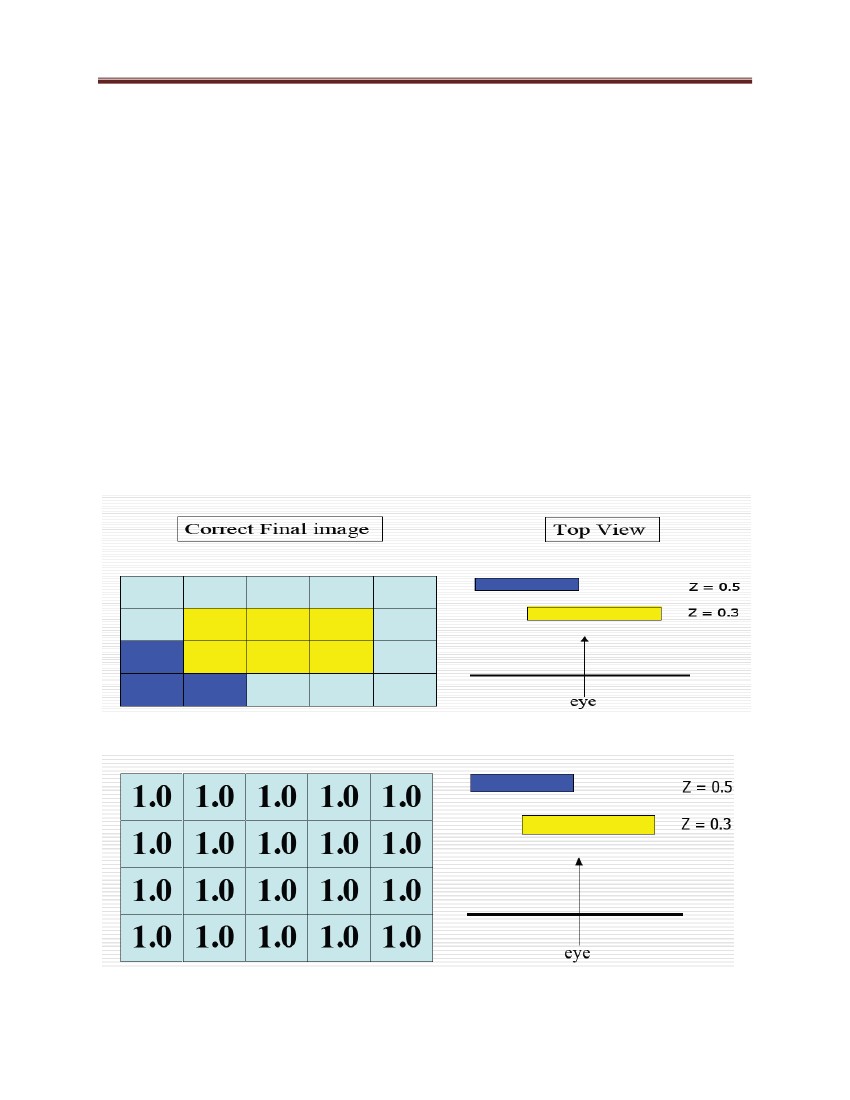
}

}

Z-Buffer Example

Step 1: Initialize the depth buffer

95



Computer Graphics and Vision

Step 2: Draw the blue polygon, assuming the program draws blue polygon first (the order

does not affect the final result anyway)

Step 3:

face, and then drawing over it

-buffer drawback: wastes resources by rendering a

b) Painter's Algorithm

Idea: Sort primitives by minimum depth, then rasterize from furthest to nearest

• When there are depth overlaps, do more tests of bounding areas, etc. to see one actually

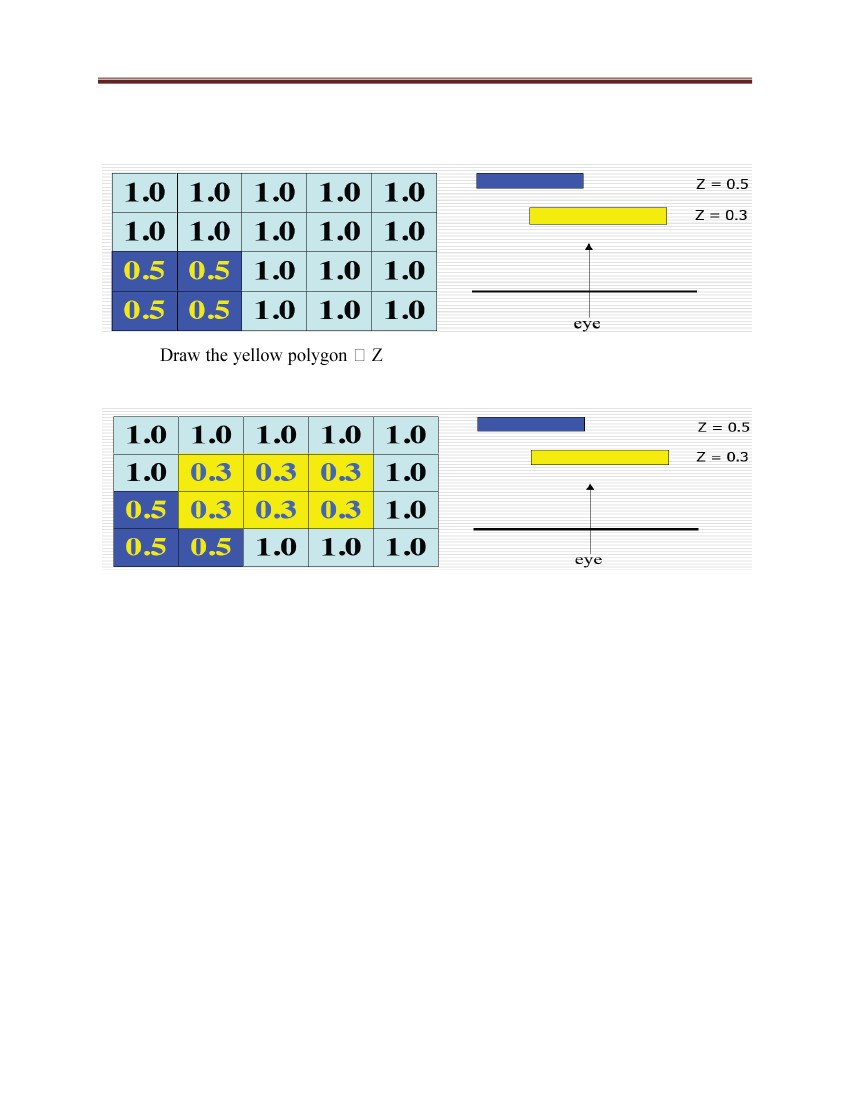
occludes the other

A depth-sorting method

Surfaces are sorted in the order of decreasing depth. Surfaces are drawn in the sorted

order, and the pixels overwritten in the frame buffer and finally the entire face drawn

96



Computer Graphics and Vision

Figure 54: Depth sorting illustration

Part 2: COMPUTER VISION

6 Principles of Computer Vision

6.1 Overview:

Computer Vision is the branch of Computer Science whose goal is to model the real world or to

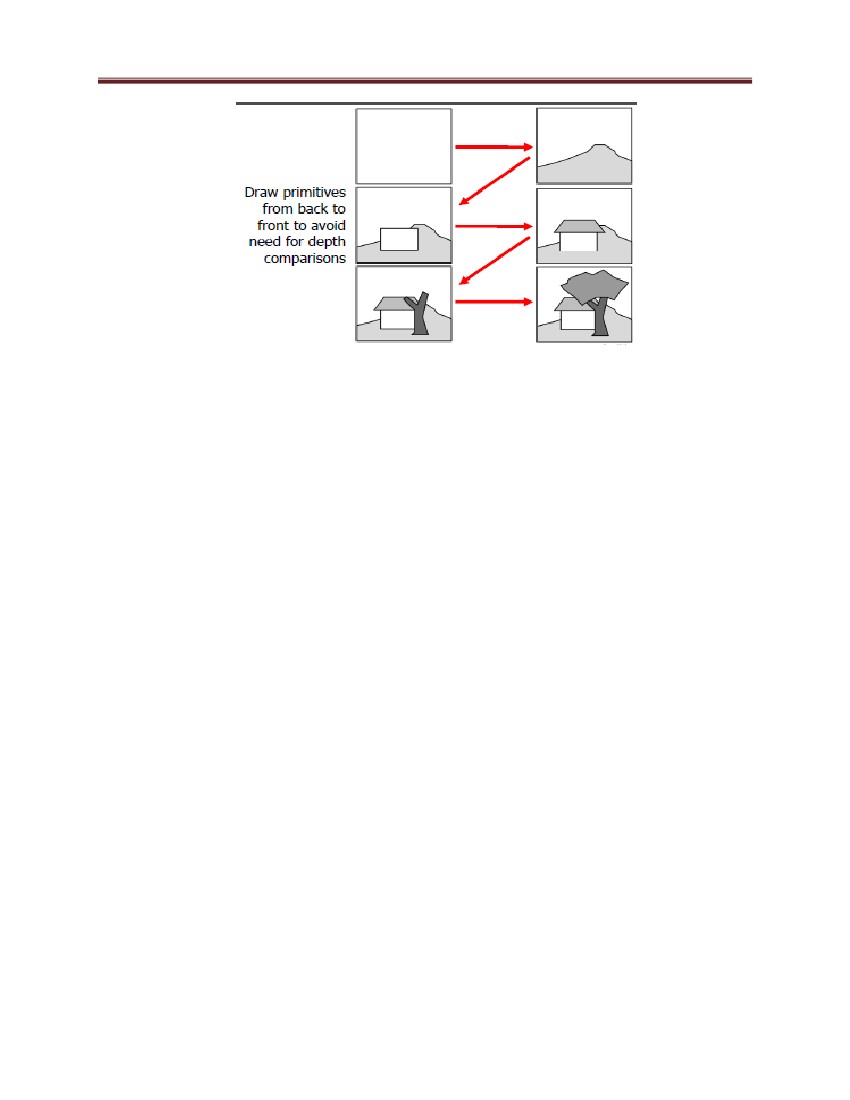
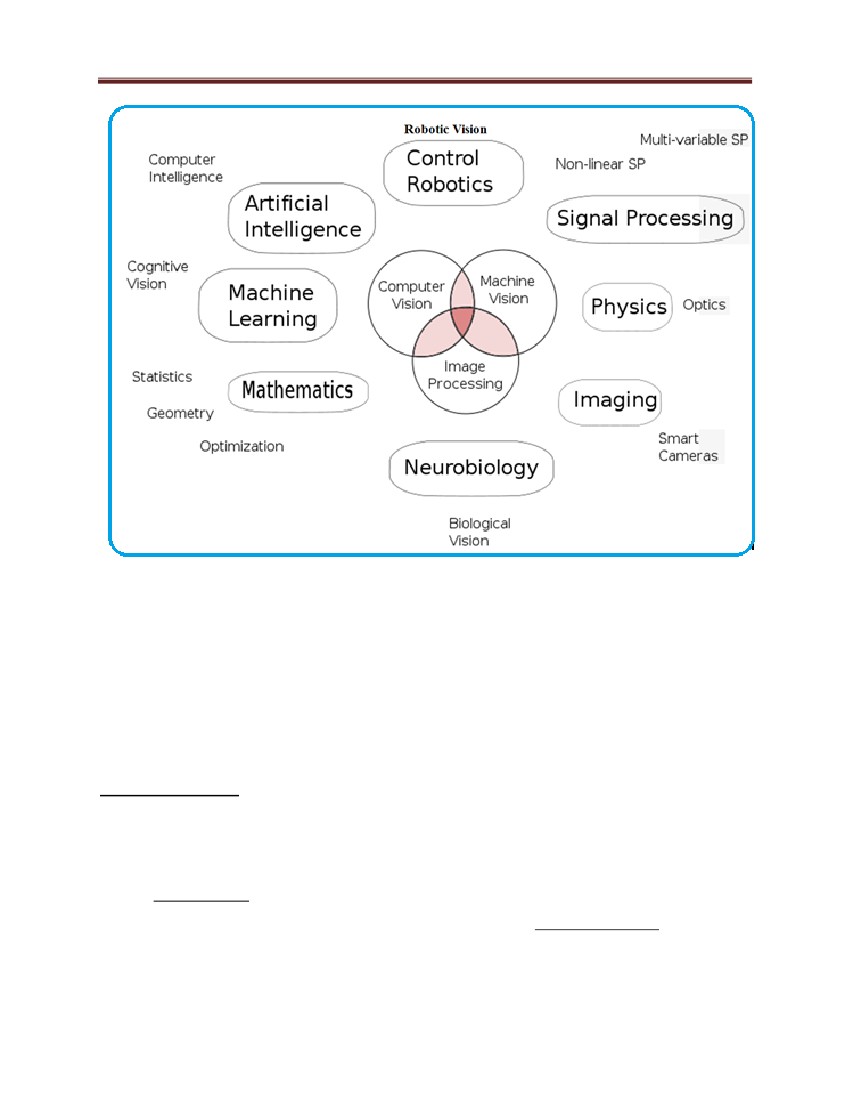
recognize objects from digital images. Computer vision is a field that includes methods for

acquiring, processing, analyzing, and understanding images and, in general, high-dimensional

data from the real world in order to produce numerical or symbolic information, e.g., in the

forms of decisions.

97



Computer Graphics and Vision

Figure 55: Relation between computer vision and various other fields

Areas of intelligence deal with autonomous planning or deliberation for robotic systems to

navigate through an environment. A detailed understanding of these environments is required to

navigate through them. Information about the environment could be provided by a computer

vision system, acting as a vision sensor and providing high-level information about the

environment and the robot. Artificial intelligence and computer vision share other topics such as

pattern recognition and learning techniques. Consequently, computer vision is sometimes seen

as a part of the artificial intelligence field or the computer science field in general.

Physics is another field that is closely related to computer vision. Most computer vision systems

rely on image sensors, which detect electromagnetic radiation which is typically in the form of

either visible or infra-red light. The sensors are designed using solid-state physics. The process

by which light interacts with surfaces is explained using physics. Physics explains the behavior

98

Computer Graphics and Vision

of optics which are a core part of most imaging systems. Also, various measurement problems in

physics can be addressed using computer vision, for example motion in fluids.

A third field which plays an important role is neurobiology, specifically the study of the

biological vision system. Over the last century, there has been an extensive study of eyes,

neurons, and the brain structures devoted to processing of visual stimuli in both humans and

various animals. This has led to a course, yet complicated, description of how "real" vision

systems operate in order to solve certain vision related tasks. These results have led to a subfield

within computer vision where artificial systems are designed to mimic the processing and

behavior of biological systems, at different levels of complexity. Also, some of the learning-

based methods developed within computer vision (e.g. neural net based image and feature

analysis and classification) have their background in biology.

Some strands of computer vision research are closely related to the study of biological vision –

indeed, just as many strands of AI research are closely tied with research into human

consciousness, and the use of stored knowledge to interpret, integrate and utilize visual

information. The field of biological vision studies and models the physiological processes behind

visual perception in humans and other animals. Computer vision, on the other hand, studies and

describes the processes implemented in software and hardware behind artificial vision systems.

Interdisciplinary exchange between biological and computer vision has proven fruitful for both

fields.

Yet another field related to computer vision is signal processing. Many methods for processing

of one-variable signals, typically temporal signals, can be extended in a natural way to

processing of two-variable signals or multi-variable signals in computer vision. However,

because of the specific nature of images there are many methods developed within computer

vision which have no counterpart in the processing of one-variable signals. Together with the

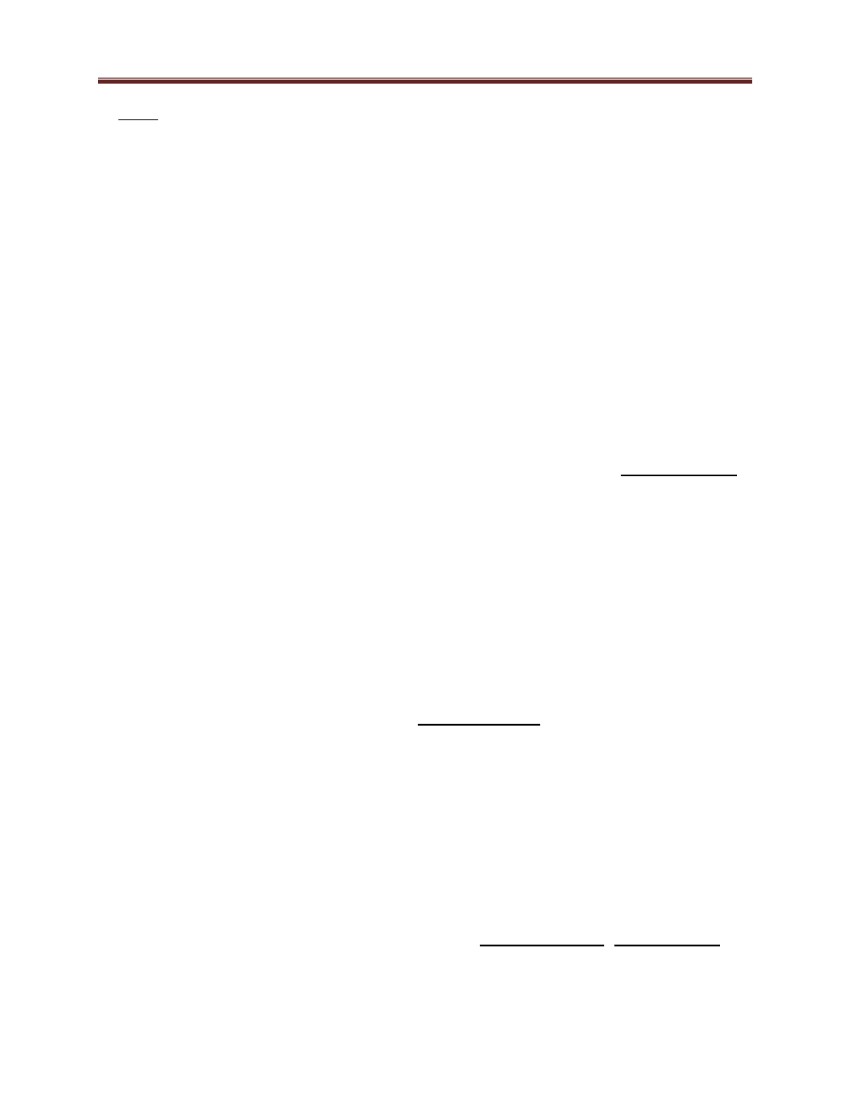
multi-dimensionality of the signal, this defines a subfield in signal processing as a part of

computer vision.

The fields most closely related to computer vision are image processing, image analysis and

machine vision. There is a significant overlap in the range of techniques and applications that

99



Computer Graphics and Vision

these cover. This implies that the basic techniques that are used and developed in these fields are

more or less identical, something which can be interpreted as there is only one field with

different names. On the other hand, it appears to be necessary for research groups, scientific

journals, conferences and companies to present or market themselves as belonging specifically to

one of these fields and, hence, various characterizations which distinguish each of the fields from

the others have been presented.

Computer vision is, in some ways, the inverse of computer graphics. While computer graphics

produces image data from 3D models, computer vision often produces 3D models from image

data.

The following characterizations appear relevant but should not be taken as universally accepted:



Image processing and image analysis tend to focus on 2D images, how to transform one

image to another, e.g., by pixel-wise operations such as contrast enhancement, local

operations such as edge extraction or noise removal, or geometrical transformations such as

rotating the image. This characterization implies that image processing/analysis neither

require assumptions nor produce interpretations about the image content.



Computer vision includes 3D analysis from 2D images. This analyzes the 3D scene projected

onto one or several images, e.g., how to reconstruct structure or other information about the

3D scene from one or several images. Computer vision often relies on more or less complex

assumptions about the scene depicted in an image.



Machine vision is the process of applying a range of technologies & methods to provide

imaging-based automatic inspection, process control and robot guidance in industrial

applications. Machine vision tends to focus on applications, mainly in manufacturing, e.g.,

vision based autonomous robots and systems for vision based inspection or measurement.

This implies that image sensor technologies and control theory often are integrated with the

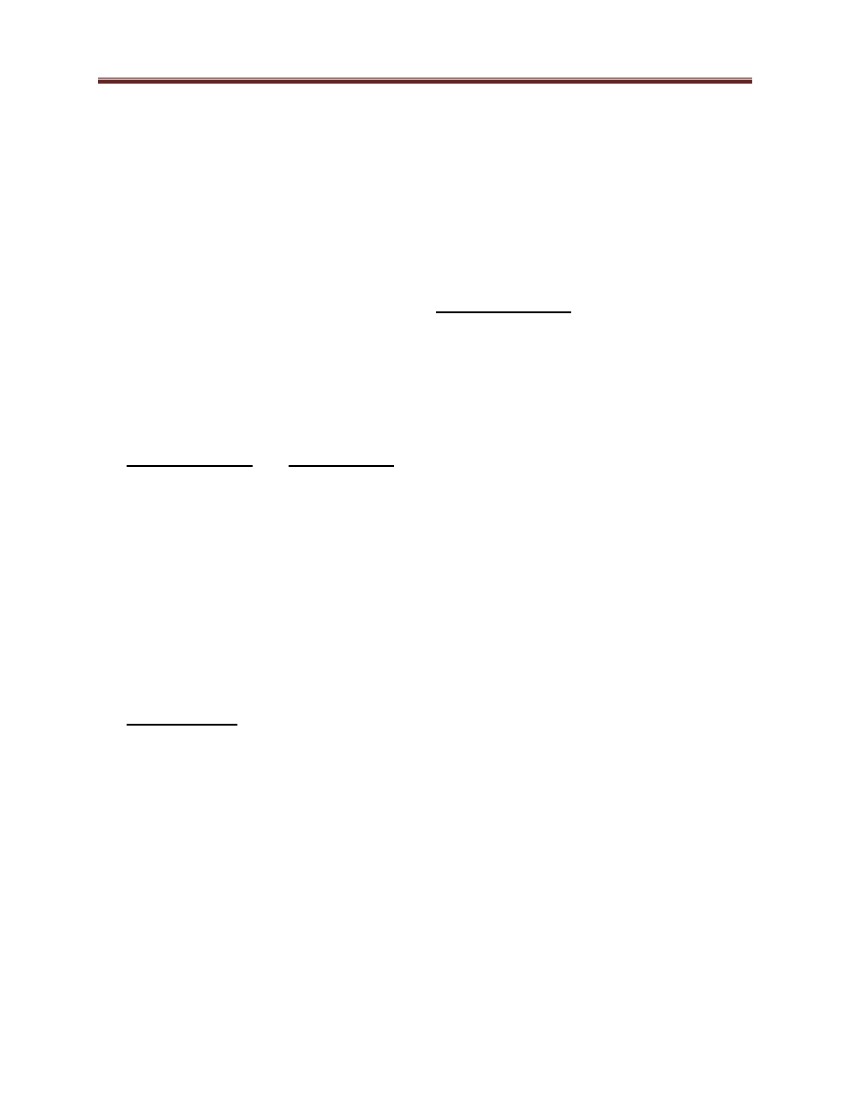
processing of image data to control a robot and that real-time processing is emphasized by

means of efficient implementations in hardware and software. It also implies that the external

conditions such as lighting can be and are often more controlled in machine vision than they

are in general computer vision, which can enable the use of different algorithms.

100



Computer Graphics and Vision



There is also a field called imaging which primarily focus on the process of producing

images, but sometimes also deals with processing and analysis of images. For example,

medical imaging includes substantial work on the analysis of image data in medical

applications.



Finally, pattern recognition is a field which uses various methods to extract information

from signals in general, mainly based on statistical approaches. A significant part of this field

is devoted to applying these methods to image data.

A theme in the development of this field has been to duplicate the abilities of human vision by

electronically perceiving and understanding an image. These images can be acquired using video

or infrared cameras, radars or specialized sensors such as those used by doctors.

Computer vision is concerned with modeling and replicating human vision using computer

software and hardware. It combines knowledge in computer science, electrical engineering,

mathematics, physiology, biology, and cognitive science. It needs knowledge from all these

fields in order to understand and simulate the operation of the human vision system. Computer

vision (image understanding) is a discipline that studies how to reconstruct, interpret and

understand a 3D scene from its 2D images in terms of the properties of the structures present in

the scene.

Figure 56: Digital Imaging

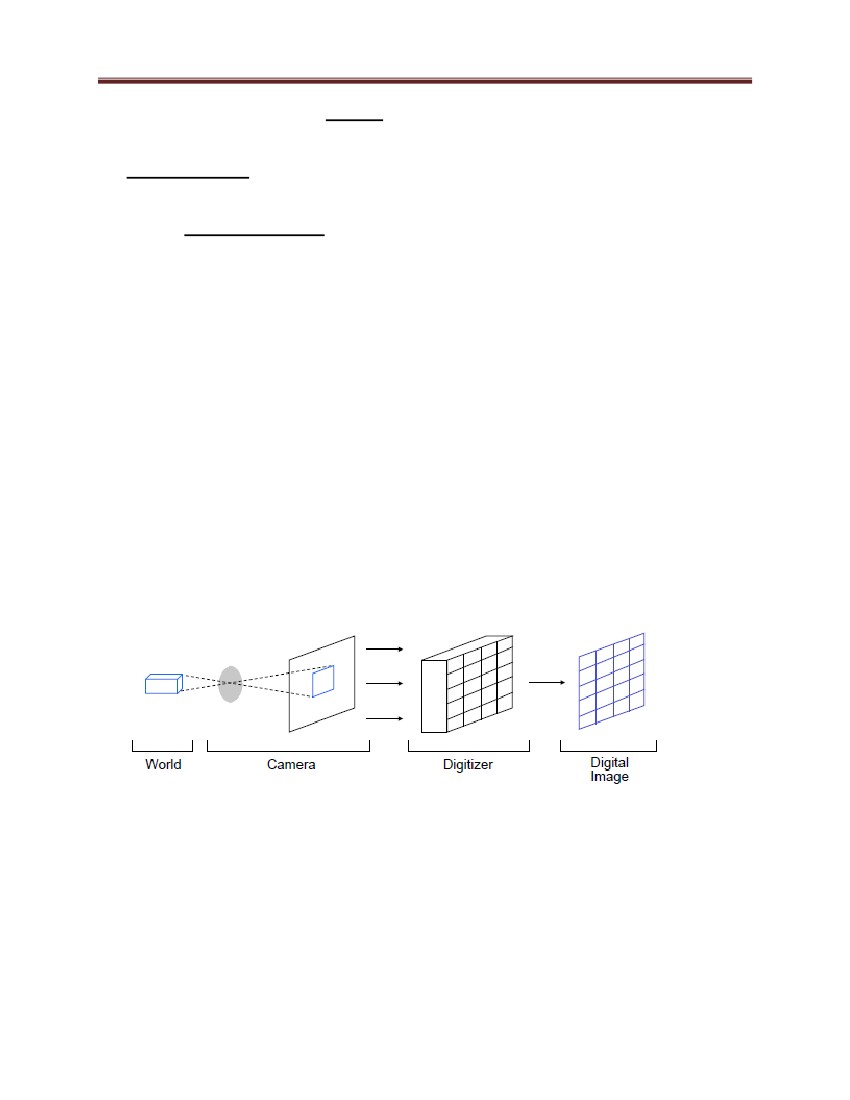
The goal of computer vision is to enable computers see and interpret the world. By using a

camera as the eye of a computer, studies in computer vision seek to develop better means to

capture and extract useful visual information from images and videos and to use such

information to automatically interpret the beautiful world surrounding us.

101



Computer Graphics and Vision

Figure 57: Typical hardware components of a computer vision system

Computer vision is sometimes considered as a separate discipline from computer graphics,

although they share many things in common.

Human Vision: Eyes are our sensors and the brain is our computer, Easy for us but we don‘t

know how we do it.

6.2 Application of Computer vision:

One of the most prominent application fields is medical computer vision or medical image

processing. This area is characterized by the extraction of information from image data for the

purpose of making a medical diagnosis of a patient. Generally, image data is in the form of

microscopy images, X-ray images, angiography images, ultrasonic images, and tomography

images. An example of information which can be extracted from such image data is detection of

tumors, arteriosclerosis or other malign changes. It can also be measurements of organ

dimensions, blood flow, etc. This application area also supports medical research by providing

new information, e.g., about the structure of the brain, or about the quality of medical treatments.

A second application area in computer vision is in industry, sometimes called machine vision,

where information is extracted for the purpose of supporting a manufacturing process. One

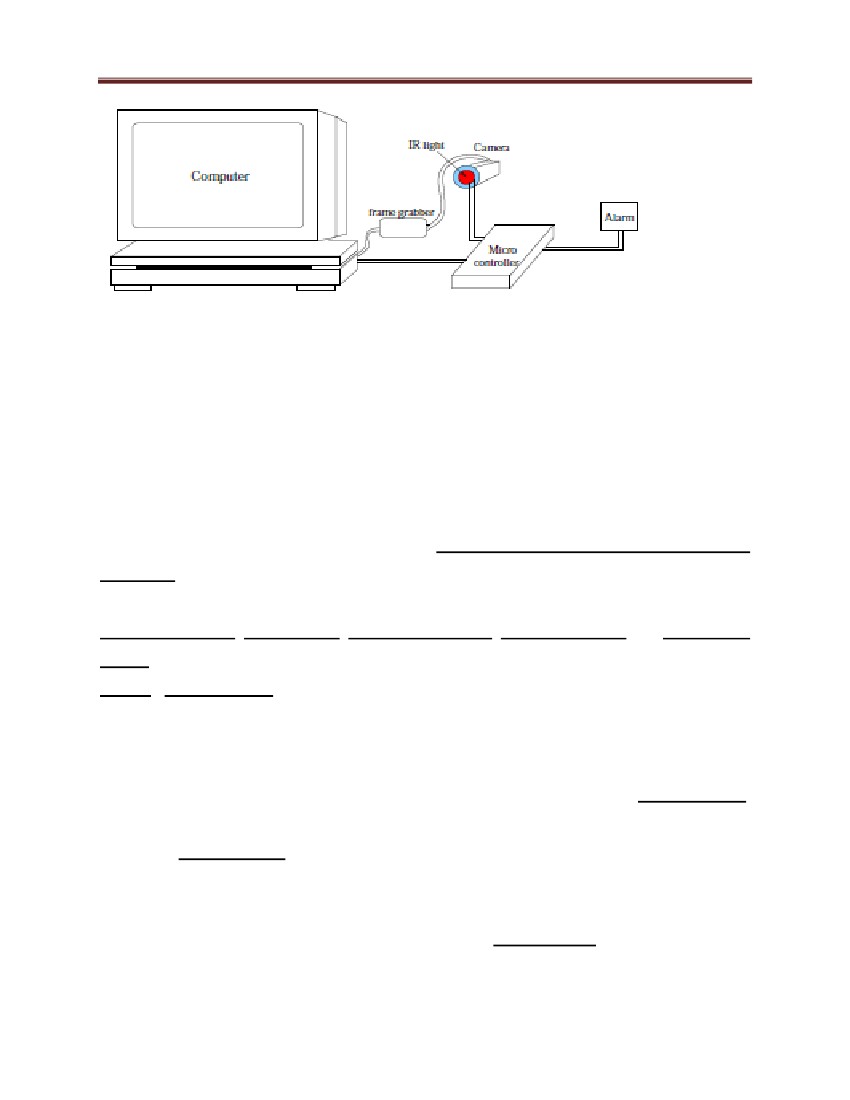
example is quality control where details or final products are being automatically inspected in

order to find defects. Another example is measurement of position and orientation of details to be

picked up by a robot arm. Machine vision is also heavily used in agricultural process to remove

undesirable food stuff from bulk material, a process called optical sorting.

102



Computer Graphics and Vision

Military applications are probably one of the largest areas for computer vision. The obvious

examples are detection of enemy soldiers or vehicles and missile guidance. More advanced

systems for missile guidance send the missile to an area rather than a specific target, and target

selection is made when the missile reaches the area based on locally acquired image data.

Modern military concepts, such as "battlefield awareness", imply that various sensors, including

image sensors, provide a rich set of information about a combat scene which can be used to

support strategic decisions. In this case, automatic processing of the data is used to reduce

complexity and to fuse information from multiple sensors to increase reliability.

In robotic applications, we can mention some tasks like:











Localization-determine robot location automatically

Obstacles avoidance

Assembly (peg-in-hole, welding, painting)

Manipulation (e.g. PUMA robot manipulator)

Intelligent robotics to interact with and serve people

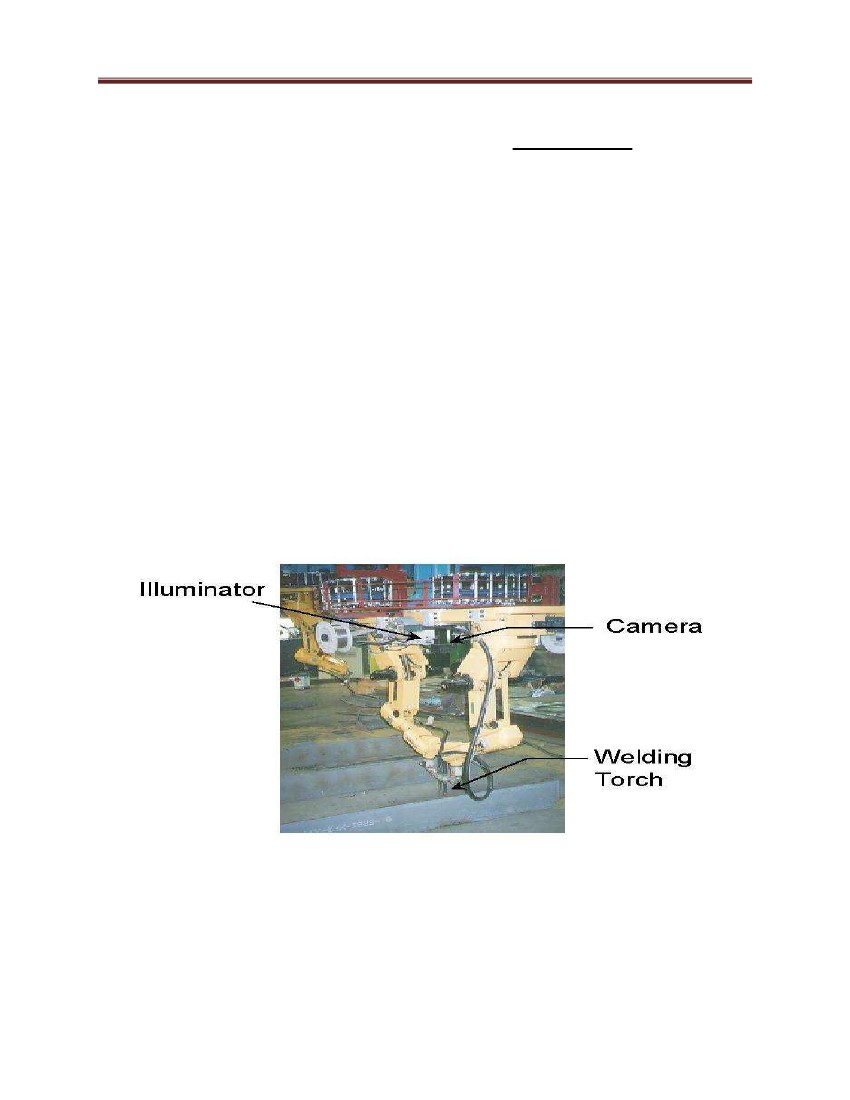
Figure 58: A vision-guided welding machine

In security Application, we can list tasks like:



Biometrics (iris, finger print, face recognition)

103



Computer Graphics and Vision





Surveillance-detecting certain suspicious activities or behaviors

Face detection

Figure 59: Face detection

Examples in transportation application are:

• Autonomous vehicle

• Safety, e.g., driver vigilance monitoring

6.3 Camera

There are many types of imaging devices, from animal eyes to video cameras and radio

telescopes. They may or may not be equipped with lenses. For example, the first models of the

camera obscura (literally, dark chamber) invented in the 16th century did not have lenses, but

instead used a pinhole to focus light rays onto a wall or translucent plate and demonstrate the

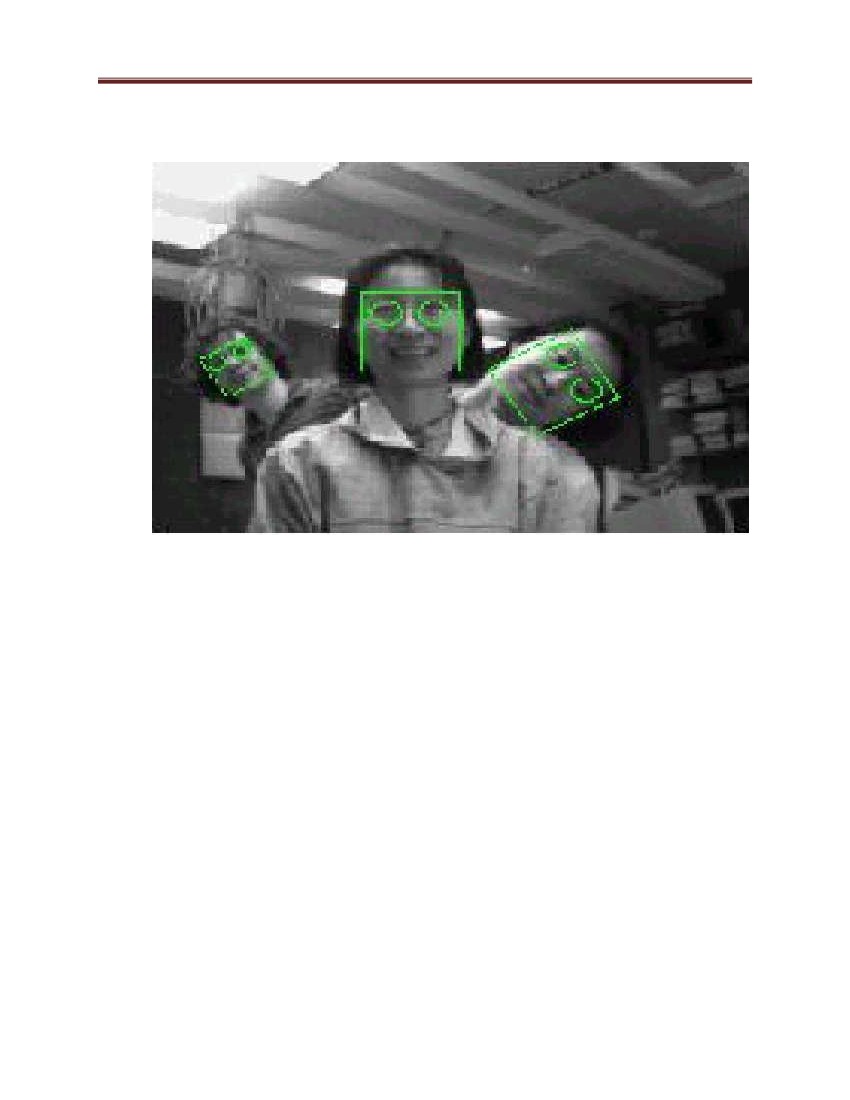
laws of perspective discovered a century earlier by Brunelleschi. Pinholes were replaced by more

and more sophisticated lenses as early as 1550, and the modern photographic or digital camera

is essentially a camera obscura capable of recording the amount of light striking every small area

of its backplane (Figure 1.1).

104



Computer Graphics and Vision

Camera = Latin for ―room‖

Obscura = Latin for ―dark‖

Figure 60: Image formation on the backplate of a photographic camera

The imaging surface of a camera is generally a rectangle, but the shape of the human retina is

much closer to a spherical surface, and panoramic cameras may be equipped with cylindrical

retinas. Imaging sensors have other characteristics. They may record a spatially discrete picture

(like our eyes with their rods and cones, 35 mm cameras with their grain, and digital cameras

with their rectangular picture elements or pixels) or a continuous one (in the case of old-

fashioned TV tubes, for example). The signal that an imaging sensor records at a point on its

retina may be discrete or continuous, and it may consist of a single number (black-and-white

camera), a few values (e.g., the R G B intensities for a color camera or the responses of the three

types of cones for the human eye), many numbers (e.g., the responses of hyperspectral sensors),

or even a continuous function of wavelength (which is essentially the case for spectrometers).

6.3.1 Pinhole cameras

a) Perspective Projection

105



Computer Graphics and Vision

Imagine taking a box, pricking a small hole in one of its sides with a pin, and then replacing the

opposite side with a translucent plate. If you hold that box in front of you in a dimly lit room,

with the pinhole facing some light source (say a candle), you see an inverted image of the candle

appearing on the translucent plate (Figure 61). This image is formed by light rays issued from the

scene facing the box. If the pinhole was really reduced to a point (which is of course physically

impossible), exactly one light ray would pass through each point in the plane of the plate (or

image plane), the pinhole, and some scene point.

Figure 61: Idealized model that defines perspective projection

• All rays go through a hole and form a star of lines

• The hole acts as a selector of rays that allows the formation of an inverted image

In reality, the pinhole has a finite (albeit small) size, and each point in the image plane collects

light from a cone of rays subtending a finite solid angle, so this idealized and extremely simple

model of the imaging geometry does not strictly apply. In addition, real cameras are normally

equipped with lenses. Still, the pinhole perspective (also called central perspective) projection

model, first proposed by Brunelleschi at the beginning of the 15th century, is mathematically

convenient. Despite its simplicity, it often provides an acceptable approximation of the imaging

process. Perspective projection creates inverted images, and it is sometimes convenient to

consider instead a virtual image associated with a plane lying in front of the pinhole at the same

distance from it as the actual image plane. See the figures No 62 .

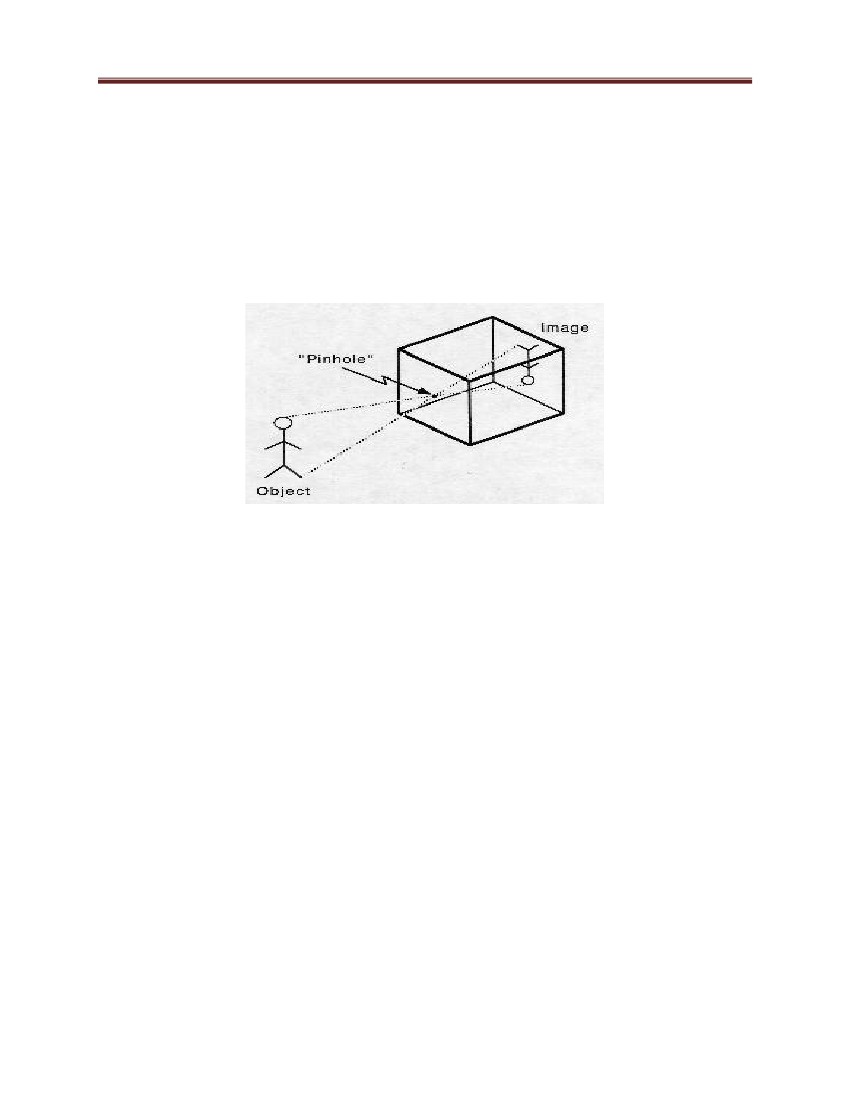
This virtual image is not inverted, but is otherwise strictly equivalent to the actual one.

Depending on the context, it may be more convenient to think about one or the other. The

apparent size of objects depends on their distance. For example, the images B‘ and C‘ of the

posts B and C have the same height, but A and C are really half the size of B.

106



Computer Graphics and Vision

Figure 62: The pinhole imaging models.

Some perspective effects:

Far objects appear smaller than close ones: the distance d from the pinhole O to the plane containing

C is half the distance from O to the plane containing A and B

Note that the image plane is behind the pinhole in (a) (physical retina), and in front of it for

virtual image plane.

These properties are easy to prove in a purely geometric fashion. However, it is often convenient

(if not quite as elegant) to reason in terms of reference frames, coordinates, and equations.

Consider, for example, a coordinate system (O, i, j, k) attached to a pinhole camera, whose origin

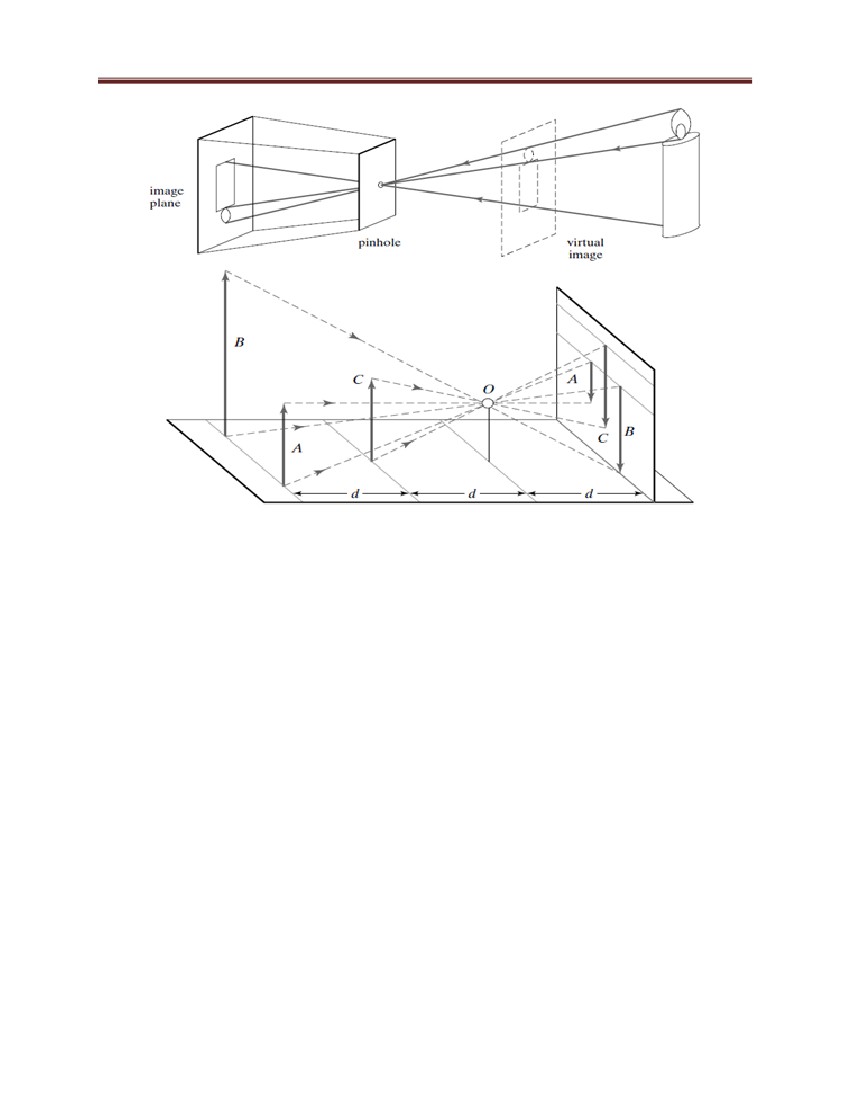
O coincides with the pinhole, and vectors i and j form a basis for a vector plane parallel to the

image plane Π‘, which is located at a positive distance f ‗ from the pinhole along the vector k

(Figure 1.4). The line perpendicular to Π‘ and passing through the pinhole is called the optical

axis, and the point C‘ where it pierces Π‘ is called the image center. This point can be used as the

107



Computer Graphics and Vision

origin of an image plane coordinate frame, and it plays an important role in camera calibration

procedures.

Let P denote a scene point with coordinates (x, y, z) and P‘ denote its image with coordinates

(x‘, y‘, z‘). Since P‘ lies in the image plane, we have z‘ = f ‗. Since the three points P, O,

Figure 63: The perspective projection equations are derived in this section from the collinearity

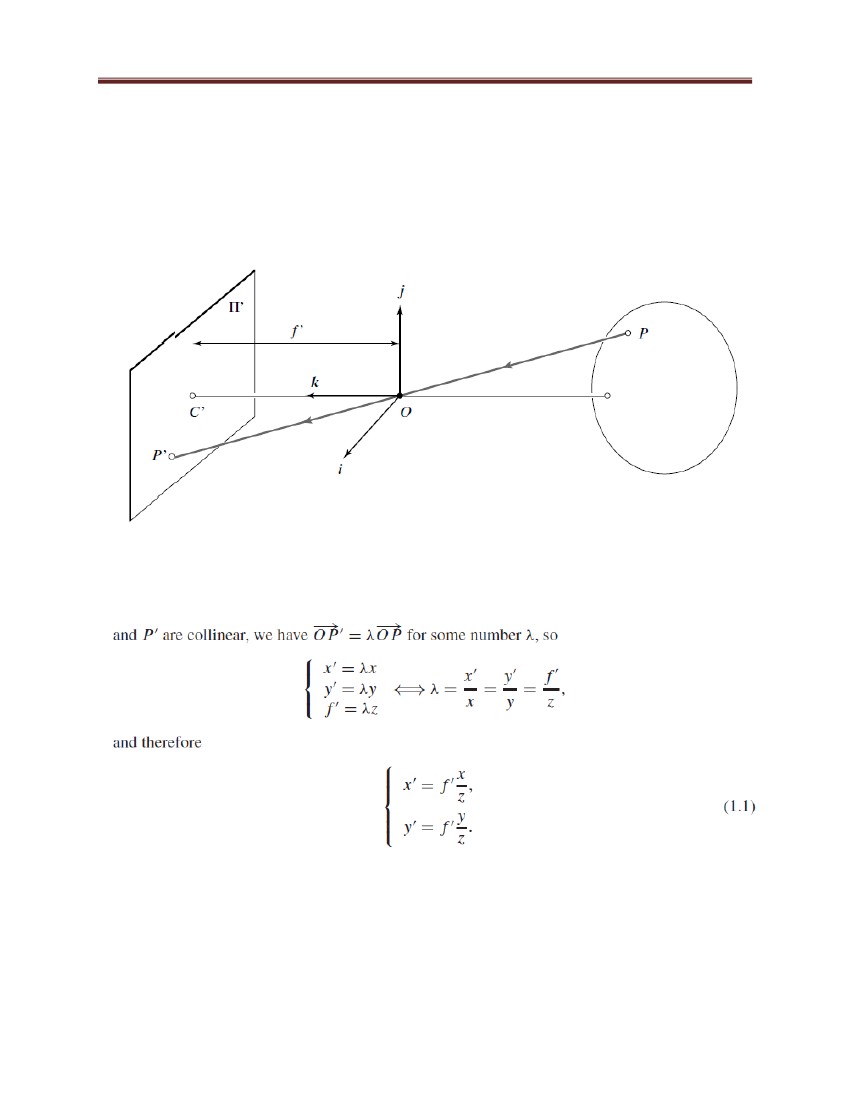
of the point P, its image P‘ , and the pinhole O.

b) Affine Projection

As noted in the previous section, pinhole perspective is only an approximation of the geometry

of the imaging process. This section discusses a class of coarser approximations, called affine

108



Computer Graphics and Vision

projection models that are also useful on occasion. We focus on two specific affine models—

namely, weak-perspective and orthographic projections. Consider the front-parallel plane Π0

defined by z = z0 (Figure 1.5). For any point P in Π0 we can rewrite the perspective projection

Eq. (1.1) as

Magnification=m

When the scene depth is small relative to the average distance from the camera, the

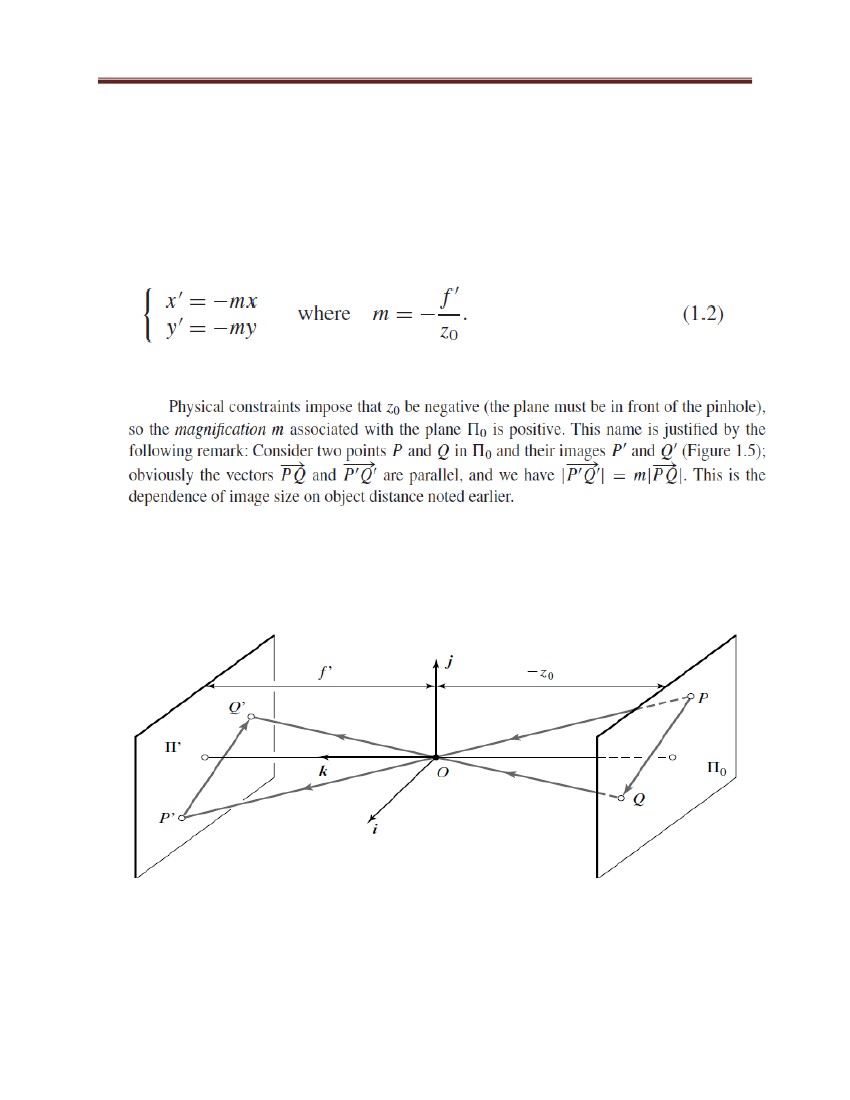
magnification can be taken to be constant. This projection model is called weak perspective or

scaled orthography.

Figure 64: Weak-perspective projection illustration.

All line segments in the plane Π0 are projected with the same magnification.

109



Computer Graphics and Vision

Figure 65: Orthographic projection.

Unlike other geometric models of the image-formation process, orthographic projection does not

involve a reversal of image features. Accordingly, the magnification is taken to be negative,

which is a bit unnatural, but simplifies the projection equations. When it is a priori known that

the camera always remains at a roughly constant distance from the scene, we can go further and

normalize the image coordinates so that m = −1.

This is orthographic projection that is defined by:

with all light rays parallel to the k axis and orthogonal to the image plane Π‘ (Figure 65).

Although weak-perspective projection is an acceptable model for many imaging conditions,

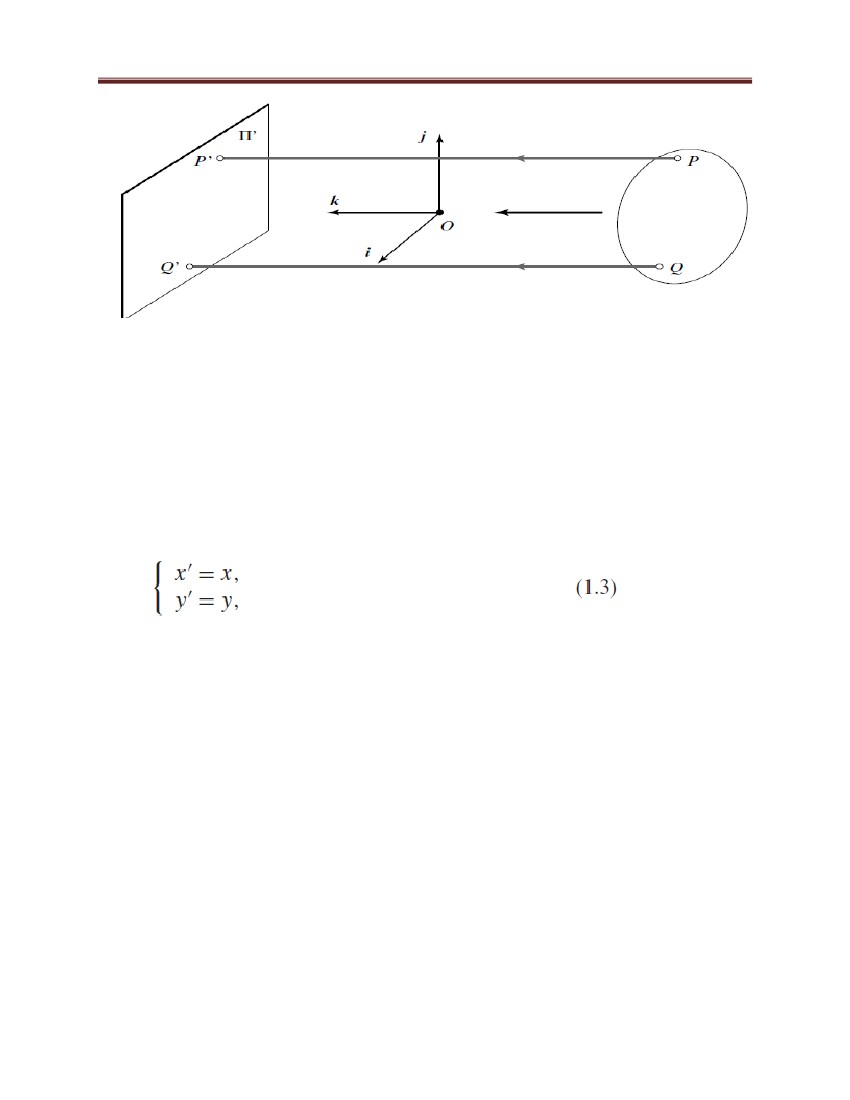
assuming pure orthographic projection is usually unrealistic.

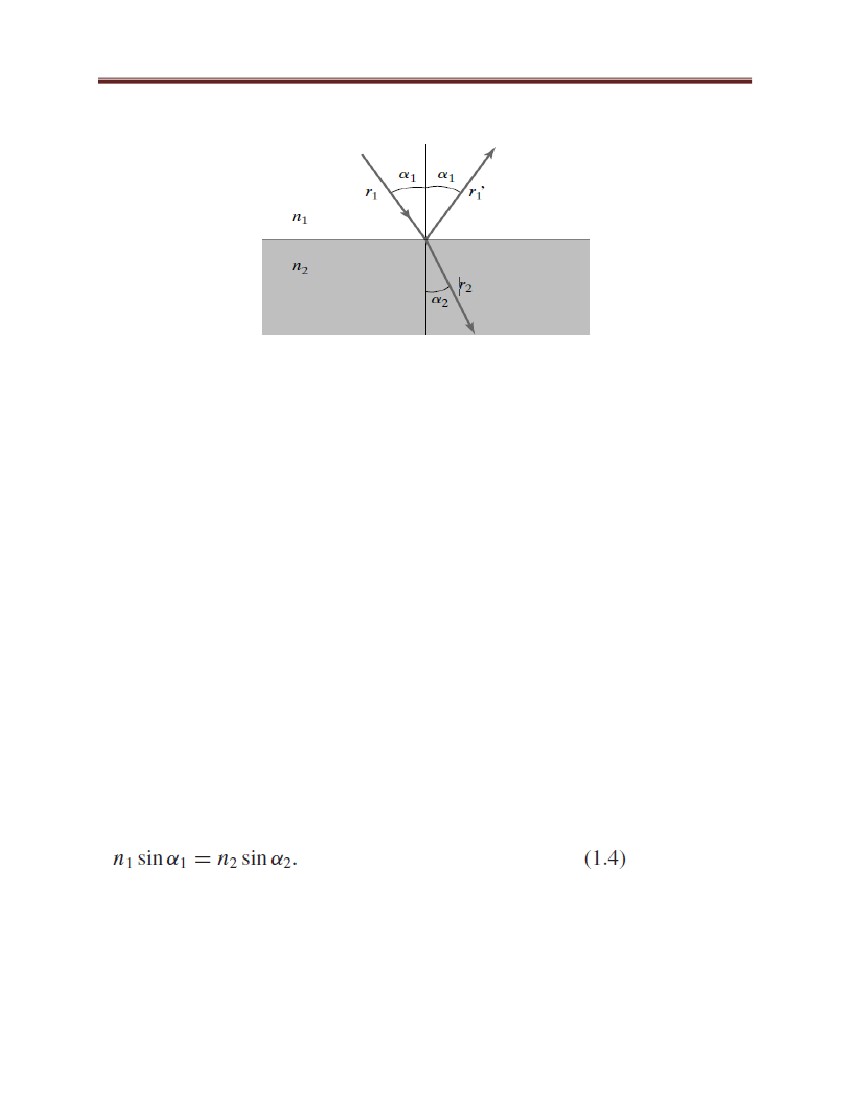
6.3.2 Cameras with Lenses

Most cameras are equipped with lenses. There are two main reasons for this: The first one is to

gather light since a single ray of light would otherwise reach each point in the image plane under

110



Computer Graphics and Vision

ideal pinhole projection. Real pinholes have a finite size of course, so each point in the image

Figure 66 Reflection and refraction

At the interface between two homogeneous media with indexes of refraction n1 and n2 plane is

illuminated by a cone of light rays subtending a finite solid angle. The larger the hole, the wider

the cone and the brighter the image, but a large pinhole gives blurry pictures. Shrinking the

pinhole produces sharper images, but reduces the amount of light reaching the image plane, and

may introduce diffraction effects. The second main reason for using a lens is to keep the picture

in sharp focus while gathering light from a large area. Ignoring diffraction, interferences, and

other physical optics phenomena, the behavior of lenses is dictated by the laws of geometric

optics (Figure 66):

(1) Light travels in straight lines (light rays) in homogeneous media;

(2) When a ray is reflected from a surface, this ray, its reflection, and the surface normal are

coplanar, and the angles between the normal and the two rays are complementary; and

(3) When a ray passes from one medium to another, it is refracted (i.e. its direction changes).

According to Snell‘s law, if r1 is the ray incident to the interface between two transparent

materials with indexes of refraction n1 and n2, and r2 is the refracted ray, then r1, r2 and the

normal to the interface are coplanar, and the angles α1 and α2 between the normal and the two

rays are related by:

In this chapter, we only consider the effects of refraction and ignore those of reflection.

In other words, we concentrate on lenses as opposed to catadioptric optical systems (e.g.,

telescopes) that may include both reflective (mirrors) and refractive elements. Tracing light rays

111

Computer Graphics and Vision

as they travel through a lens is simpler when the angles between these rays and the refracting

surfaces of the lens are assumed to be small. The next section discusses this case.

6.3.3 Paraxial Geometric Optics

In this section, we consider paraxial (or first-order) geometric optics, where the angles between

all light rays going through a lens and the normal to the refractive surfaces of the lens are small.

In addition, we assume that the lens is rotationally symmetric about a straight line, called its

optical axis, and that all refractive surfaces are spherical. The symmetry of this setup allows us

to determine the projection geometry by considering lenses with circular boundaries lying in a

plane that contains the optical axis.

Let us consider an incident light ray passing through a point P1 on the optical axis and

refracted at the point P of the circular interface of radius R separating two transparent media

with indexes of refraction n1 and n2 (Figure 67). Let us also denote by P2 the point where the

refracted ray intersects the optical axis a second time (the roles of P1 and P2 are completely

symmetric) and by C the center of the circular interface

Figure 67: Paraxial refraction

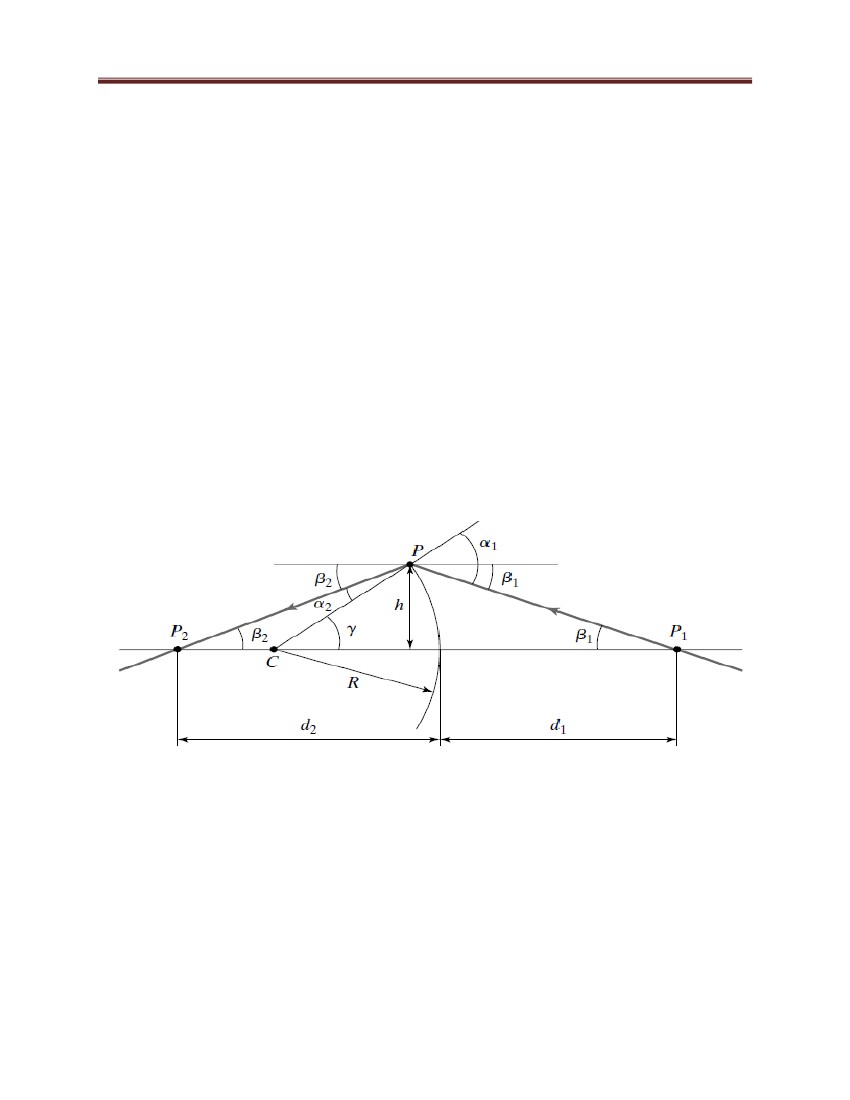
A light ray passing through the point P1 is refracted at the point P where it intersects a circular

interface. The refracted ray intersects the optical axis in P2. The center of the interface is at the

point C of the optical axis, and its radius is R. The angles α1, β1, α2, and β2 are all assumed to

be small

112



Computer Graphics and Vision

Let α1 and α2, respectively, denote the angles between the two rays and the chord joining C to P.

If β1 (resp. β2) is the angle between the optical axis and the line joining P1 (resp. P2) to P, the

angle between the optical axis and the line joining C to P is, as shown by Figure 67, γ = α1 −β1 =

α2 +β2. Now let h denote the distance between P and the optical axis and R the radius of the

circular interface. If we assume all angles are small and thus, to first order, equal to their sines

and tangents, we have:

Note that the relationship between d1 and d2 depends on R, n1, and n2, but not on β1 or β2. This is

the main simplification introduced by the paraxial assumption. It is easy to see that Eq. (1.5)

remains valid when some (or all) of the values of d1, d2, and R become negative, corresponding

to the points P1, P2, or C switching sides. Of course, real lenses are bounded by at least two

refractive surfaces. The corresponding ray paths can be constructed iteratively using the paraxial

refraction equation.

6.3.4 Thin Lenses

Let us now consider a lens with two spherical surfaces of radius R and index of refraction n. We

assume that this lens is surrounded by vacuum (or, to an excellent approximation, by air), with

an index of refraction equal to 1, and that it is thin (i.e., that a ray entering the lens and refracted

at its right boundary is immediately refracted again at the left boundary). Consider a point P

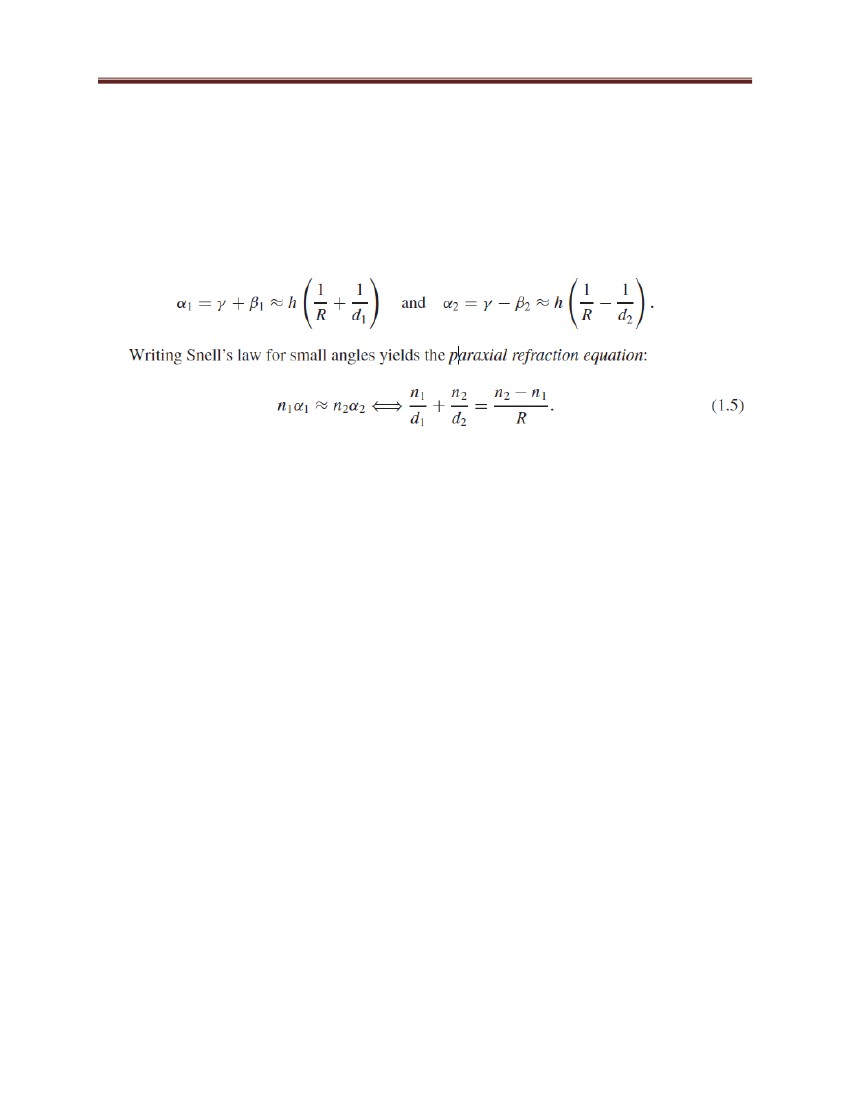
located at (negative) depth z off the optical axis and denote by (PO) the ray passing through this

point and the center O of the lens (Figure 1.9). As shown in the exercises, it follows from Snell‘s

law and Eq. (1.5) that the ray (PO) is not refracted and that all other rays passing through P are

focused by the thin lens on the point P‘ with depth z‘ along

113



Computer Graphics and Vision

Figure 68: A thin lens. Rays passing through the point O are not refracted. Rays parallel to the

optical axis are focused on the focal point F‘

Note that the equations relating the positions of P and P‘ are exactly the same as under pinhole

perspective projection if we take z‘ = f ‗, since P and P‘ lie on a ray passing through the center of

the lens, but that points located at a distance −z from O are only in sharp focus when the image

plane is located at a distance z‘ from O on the other side of the lens that satisfies Eq. (1.6) (i.e.,

the thin lens equation). Letting z → −∞shows that f is the distance between the center of the lens

and the plane where objects such as stars, which are effectively located at z = −∞, focus.

The two points F and F‘ located at distance f from the lens center on the optical axis are called

the focal points of the lens.

In practice, objects within some range of distances (called depth of field or depth of focus) are in

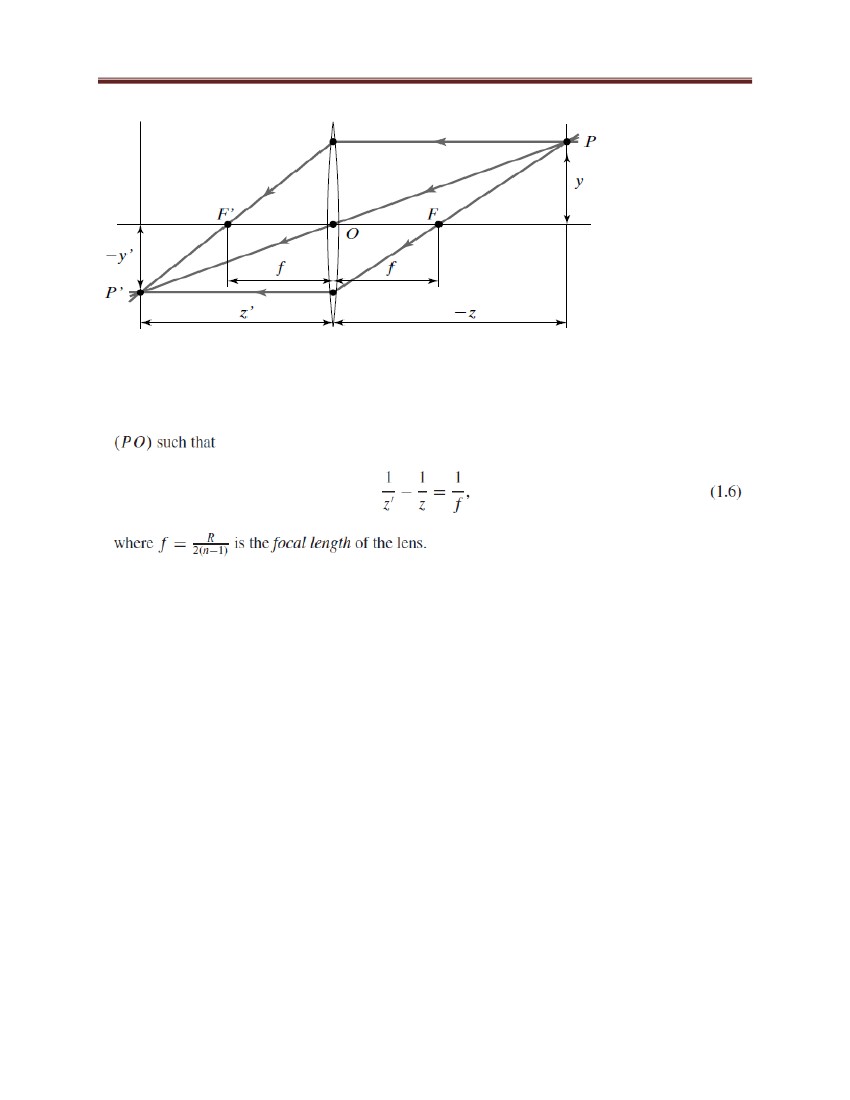
acceptable focus. As shown in the exercises, the depth of field increases with the f number of the

lens (i.e., the ratio between the focal length of the lens and its diameter). The field of view of a

camera is the portion of scene space that actually projects onto the retina of the camera. It is not

defined by the focal length alone, but also depends on the effective area of the retina.

114



Computer Graphics and Vision

6.3.5 The human

eye

It is largely based on the presentation in Wandell (1995).

Figure 69 (left) is a sketch of the section of an eyeball through its vertical plane of symmetry,

showing the main elements of the eye: the iris and the pupil, which control the amount of light

penetrating the eyeball; the cornea and the crystalline lens, which together refract the light to

create the retinal image; and finally the retina, where the image is formed.

Despite its globular shape, the human eyeball is functionally similar to a camera with a field of

view covering a 160◦ (width) × 135◦ (height) area.

Figure 69 Left: The main components of the human eye; ---Right: Helmoltz‘s schematic eye.

The distance between the pole of the cornea and the anterior principal plane is 1.96 mm, and the

radii of the cornea, anterior, and posterior surfaces of the lens are respectively 8 mm, 10 mm, and

6 mm. Like any other optical system, it suffers from various types of geometric and chromatic

aberrations. Several models of the eye obeying the laws of first-order geometric optics have been

proposed, and Figure 69 (right) shows one of them, Helmoltz‘s schematic eye as modified by

Laurance (after Driscoll and Vaughan, 1978).

There are only three refractive surfaces, with an infinitely thin cornea and a homogeneous lens.

The constants given in Figure 1.14 are for the eye focusing at infinity (unaccommodated eye).

This model is of course only an approximation of the real optical characteristics of the eye.

6.3.6 Sensing

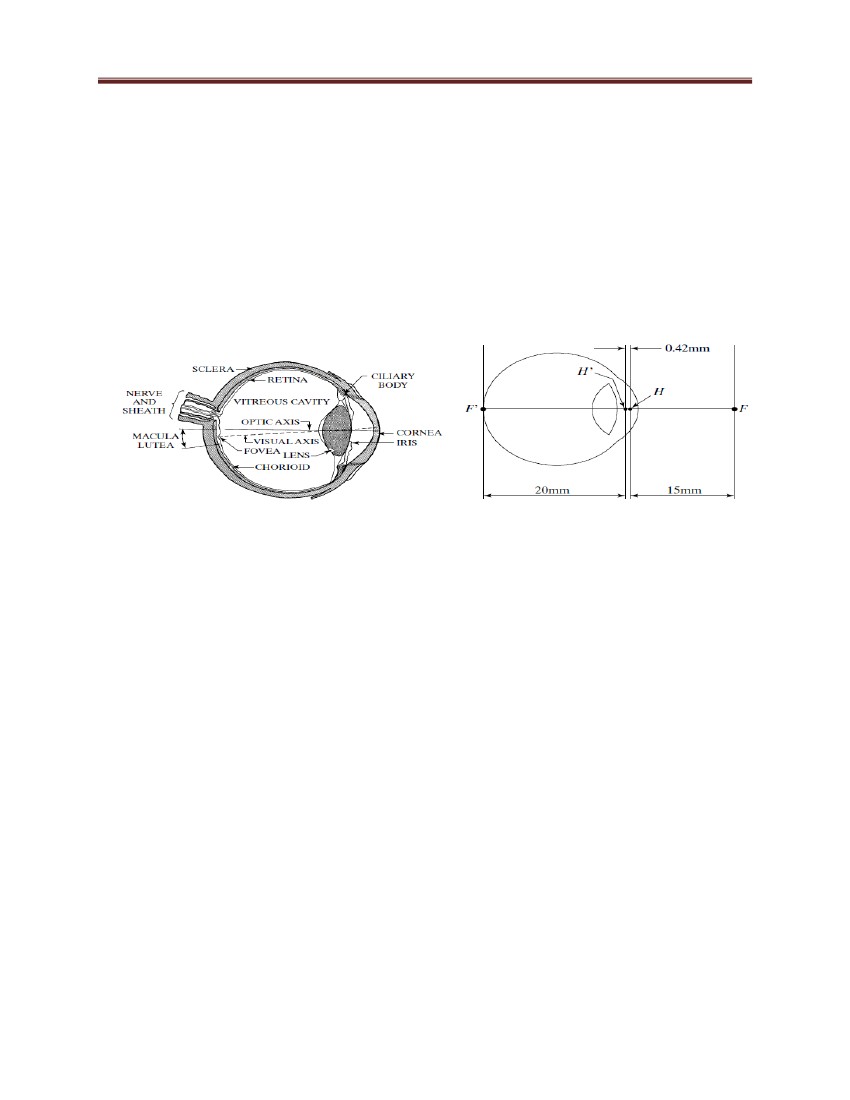
What differentiates a camera (in the modern sense of the world) from the portable camera

obscura of the 17th century is its ability to record the pictures that form on its backplane.

Although it had been known since at least the Middle Ages that certain silver salts rapidly darken

under the action of sunlight, it was only in 1816 that Niepce obtained the first true photographs

115



Computer Graphics and Vision

by exposing paper treated with silver chloride to the light rays striking the image plane of a

camera obscura, then fixing the picture with nitric acid. These first images were negatives, and

Niepce soon switched to other photosensitive chemicals to obtain positive pictures. The earliest

photographs have been lost, and the first one to have been preserved is la table servie (the set

table) obtained by Nic´ephore Niepce in 1822. Collection Harlinge–Viollet as shown in Figure

70

Figure 70: The first photograph on record, la table servie,

The invention of television in the 1920s by people like Baird, Farnsworth, and Zworykin was of

course a major impetus for the development of electronic sensors. The vidicon is a common type

of TV vacuum tube. It is a glass envelope with an electron gun at one end and a faceplate at the

other. The back of the faceplate is coated with a thin layer of photoconductor material laid over a

transparent film of positively charged metal. This double coating forms the target. The tube is

surrounded by focusing and deflecting coils that are used to repeatedly scan the target with the

electron beam generated by the gun. This beam deposits a layer of electrons on the target to

balance its positive charge. When a small area of the faceplate is struck by light, electrons flow

through, locally depleting the charge of the target. As the electron beam scans this area, it

replaces the lost electrons, creating a current proportional to the incident light intensity.

The current variations are then transformed into a video signal by the vidicon circuitry.

6.4 CCD Cameras

Let us now turn to charge-coupled-device (CCD) cameras that were proposed in 1970 and have

116



Computer Graphics and Vision

replaced vidicon cameras in most modern applications, from consumer camcorders to special

purpose cameras geared toward microscopy or astronomy applications. A CCD sensor uses a

rectangular grid of electron-collection sites laid over a thin silicon wafer to record a measure of

the amount of light energy reaching each of them (Figure 1.17).

Each site is formed by growing a layer of silicon dioxide on the wafer and then depositing a

conductive gate structure over the dioxide. When photons strike the silicon, electron-hole pairs

are generated (photo-conversion), and the electron are captured by the potential well formed by

applying a positive electrical potential to the corresponding gate. The electrons generated at each

site are collected over a fixed period of time T .

Figure 71: A CCD Device.

At this point, the charges stored at the individual sites are moved using charge coupling:

Charge packets are transferred from site to site by manipulating the gate potentials, preserving

the separation of the packets. The image is read out of the CCD one row at a time, each row

being transferred in parallel to a serial output register with one element in each column. Between

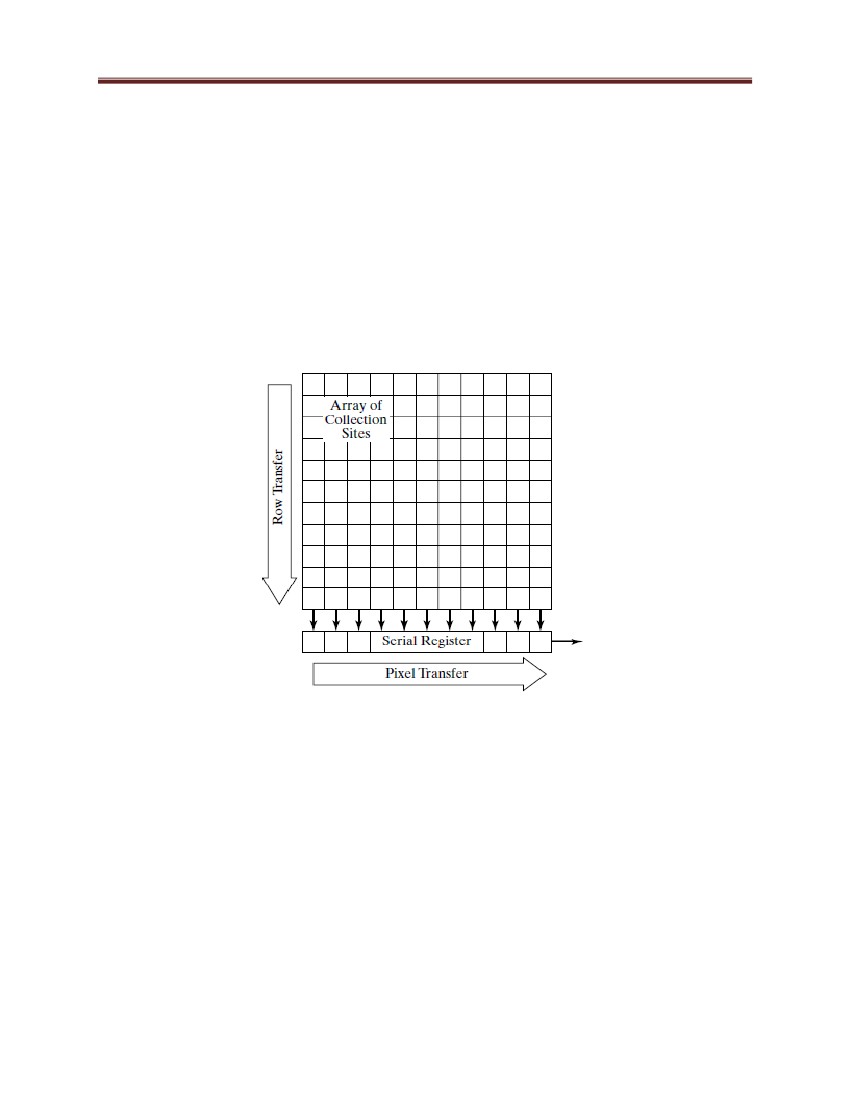
two row reads, the register transfers its charges one at a time to an output amplifier that generates

a signal proportional to the charge it receives. This process continues until the entire image has

been read out. It can be repeated 30 times per second (TV rate) for video applications or at a

much slower pace, leaving ample time (seconds, minutes, even hours) for electron collection in

117



Computer Graphics and Vision

low-light-level applications such as astronomy. It should be noted that the digital output of most

CCD cameras is transformed internally into an analog video signal before being passed to a

frame grabber that constructs the final digital image.

Table 1: Reference card: Camera models.

6.5 Source, Shadow and Shading

This sub-chapter is going to discuss about the sources of light, the shadows, and about shading.

Surfaces are bright or dark for two main reasons: their albedo and the amount of light they are

receiving.

A model of how the brightness of a surface is obtained is usually called a shading model.

Shading models are important because with an appropriate shading model we can interpret pixel

values. If the right shading model applies, it is possible to reconstruct objects and their albedos

using just a few images. Furthermore, we can interpret shadows and explain their puzzling and

seldom-noticed absence in most indoor scenes.

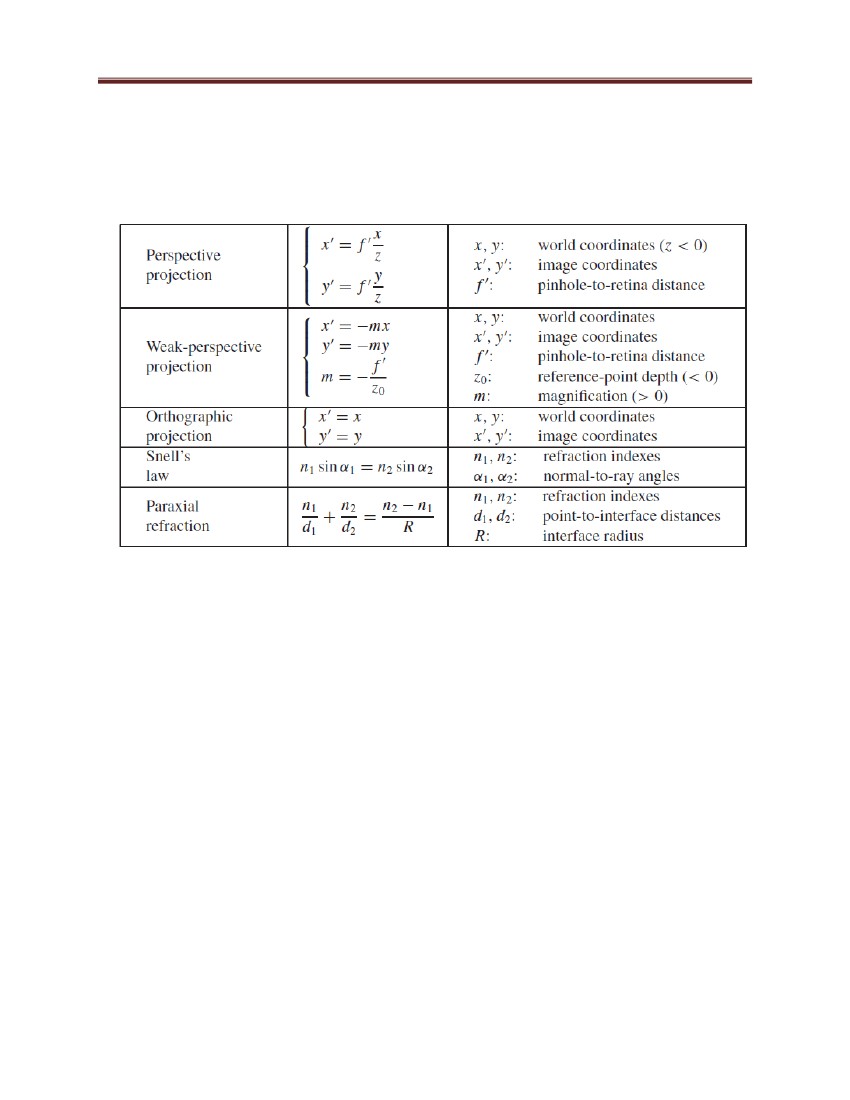
6.5.1 Qualitative radiometry

We should like to know how ―bright‖ surfaces are going to be under various lighting conditions,

and how this ―brightness‖ depends on local surface properties, on surface shape, and on

illumination. The most powerful tool for analyzing this problem is to think about what a source

118



Computer Graphics and Vision

looks like from the surface. This qualitative radiometry is one of these tricks that looks

unsophisticated—no hard math—but is extremely powerful. In some cases, this technique gives

qualitative descriptions of ―brightness‖ without even knowing what the term means.

a surface patch sees the world through a hemisphere of directions at that patch. The radiation

arriving at the surface along a particular direction passes through a point on the hemisphere.

6.5.2 Brightness sources and their effects

(a) Radiometric Properties of Light Sources

We define a light source to be anything that emits light that is internally generated (i.e., not just

reflected). To describe a source, we need a description of the radiance it emits in each direction.

Typically, internally generated radiance is dealt with separately from reflected radiance. This is

because, although a source may reflect light, the light it reflects depends on the environment,

whereas the light it generates internally usually does not.

Together with a description of the existence, we need a description of the geometry of the source,

which has profound effects on the spatial variation of light around the source and on the shadows

cast by objects near the source. Sources are usually modeled with quite simple geometries for

two reasons: first, many synthetic sources can be modeled as point sources, line sources, or area

sources fairly effectively; second, sources with simple geometries can still yield surprisingly

complex effects.

(b) Point Sources

A common approximation is to assume that the light source is an extremely small sphere, in fact,

a point; such a source is known as a point source. It is a natural model to use because many

sources are physically small compared with the environment in which they stand. We can obtain

a model for the effects of a point source by modeling the source as a very small sphere that emits

light at each point on the sphere, with an existence constant over the sphere.

(c) Line Sources

A line source has the geometry of a line—a good example is a single fluorescent light bulb. Line

sources are not terribly common in natural scenes or in synthetic environments, and we discuss

them only briefly. Their main interest is as an example for radiometric problems; in particular,

119



Computer Graphics and Vision

the radiosity of patches reasonably close to a line source changes as the reciprocal of distance to

the source (rather than the square of the distance). The reasoning is more interesting than the

effect. We model a line source as a thin cylinder with diameter ε. Assume for the moment that

the line source is infinitely long and that we are considering a patch that views the source

frontally.

(d) Area Sources

An area source is an area that radiates light. Area sources are important for two reasons. First,

they occur quite commonly in natural scenes—an overcast sky is a good example—and in

synthetic environments—for example, the fluorescent light boxes found in many industrial

ceilings. Second, a study of area sources allows us to explain various shadowing and

interreflection effects. Area sources are normally modeled as surface patches whose emitted

radiance is independent of position and of direction—they can be described by their existence.

6.5.3 Local shading Models

We have studied the physics of light because we want to know how bright things will be, and

why, in the hope of extracting object information from these models. Currently, we know the

radiosity at a patch due to a source but this is not a shading model. Radiance could arrive at

surface patches in other ways (e.g., it could be reflected from other surface patches); we need to

know which components to account for. The easiest model to manipulate is a local shading

model, which models the radiosity at a surface patch as the sum of the radiosity due only to light

internally generated at sources.

This means that we assume that light is not reflected from surface to surface, but instead leaves

a source, arrives at some surface, and proceeds directly to the camera. This model is palpably

unphysical, but is easy to analyze.

The Appearance of Shadows In a local shading model, shadows occur when the patch

cannot see one or more sources. In this model, point sources produce a series of shadows

with crisp boundaries; shadow regions where no source can be seen are particularly dark.

Shadows cast with a single source can be crisp and black depending on the size of the

source and the albedo of other nearby surfaces (which could reflect light into the shadow

and soften its boundary). It was a popular 19th Century pastime to cast such shadows

120

Computer Graphics and Vision

onto paper and then draw them, yielding the silhouettes still occasionally found in

antiques shops.

The geometry of the shadow cast by a point source on a plane is analogous to the

geometry of viewing in a perspective camera (Figure 5.6). Any patch on the plane is in

shadow if a ray from the patch to the source passes through an object. This means that

there are two kinds of shadow boundary. At self shadow boundaries, the surface is

turning away from the light, and a ray from the patch to the source is tangent to the

surface. At cast shadow boundaries, from the perspective of the patch, the source

suddenly disappears behind an occluding object.

6.6 Color

(a) What is a Color?

One definition says that a color is a distribution of energies amongst frequencies in the visible

light range. In other words, Color is the visual perceptual property corresponding in humans to

the categories called red, green, blue, and others. Color derives from the spectrum of light

(distribution of light power versus wavelength) interacting in the eye with the spectral

sensitivities of the light receptors. Color categories and physical specifications of color are also

associated with objects, materials, light sources, etc., based on their physical properties such as

light absorption, reflection, or emission spectra.

To better understand what color is, we first need to understand what light is. Light, as perceived

by humans, is simply electromagnetic radiation with wavelengths between roughly 380 nm and

740 nm. Wavelengths below 380 nm and above 740 nm cannot be seem by the human eye.

Electromagnetic radiation with a wavelength just below 380 nm is known as ultraviolet radiation.

Electromagnetic radiation with a wavelength just above 740 nm is known as infrared radiation.

The sun, black lights and fluorescent lamps are all sources of ultraviolet light. Heat is a source of

infrared radiation, which is how thermal vision works.

Electromagnetic radiation between the wavelengths of 380 nm and 740 nm constitute light and

the human color-vision spectrum. There is evidence that other animals, namely insects and birds,

have a color-vision spectrum that extends further into the ultraviolet range. Evolutionary factors

121



Computer Graphics and Vision

greatly influence the development of color vision in different animals and species.

Although the human color-vision spectrum is continuous, it can be broken down into discrete

ranges of colors which are then labeled with specific names. Here is a simplistic chart of

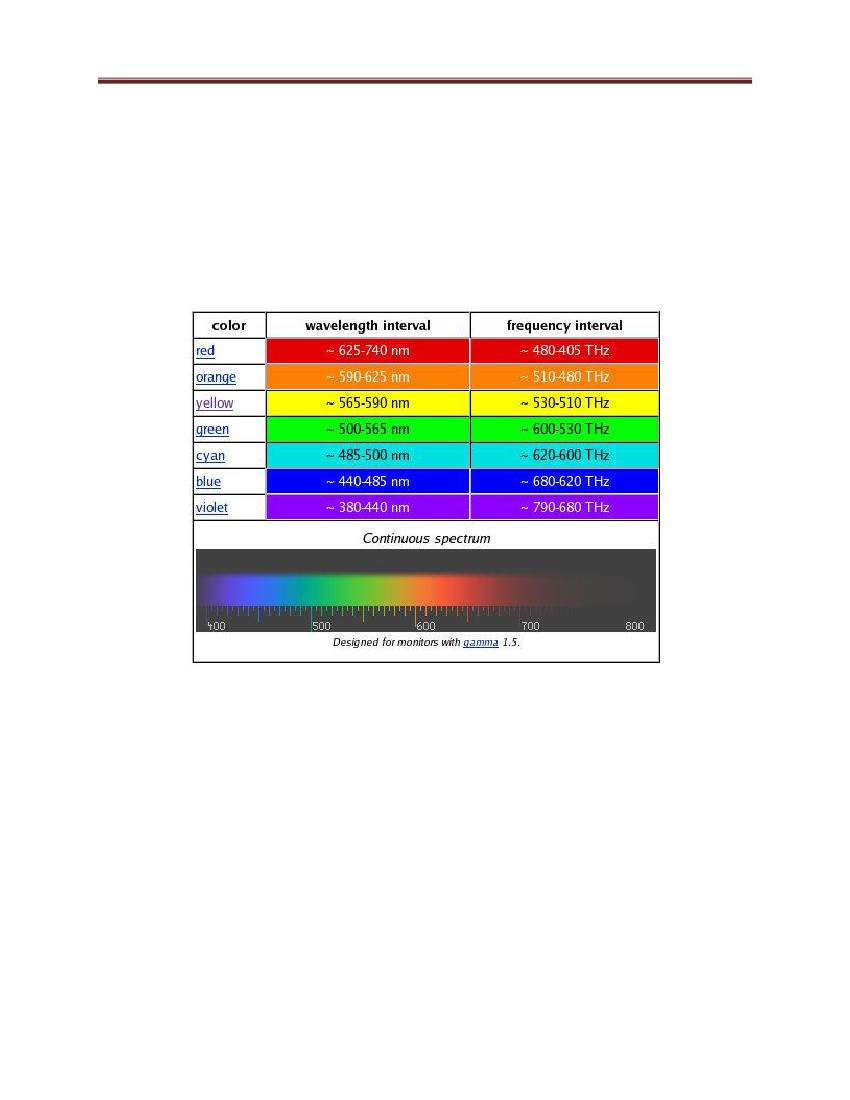
electromagnetic wavelengths and their associated color names:

Table: The colors of the visible light spectrum

(b) Representing Colors in a Computer

Common color models are: HLS, HSV, RGB, XYZ, CMY, Others

122

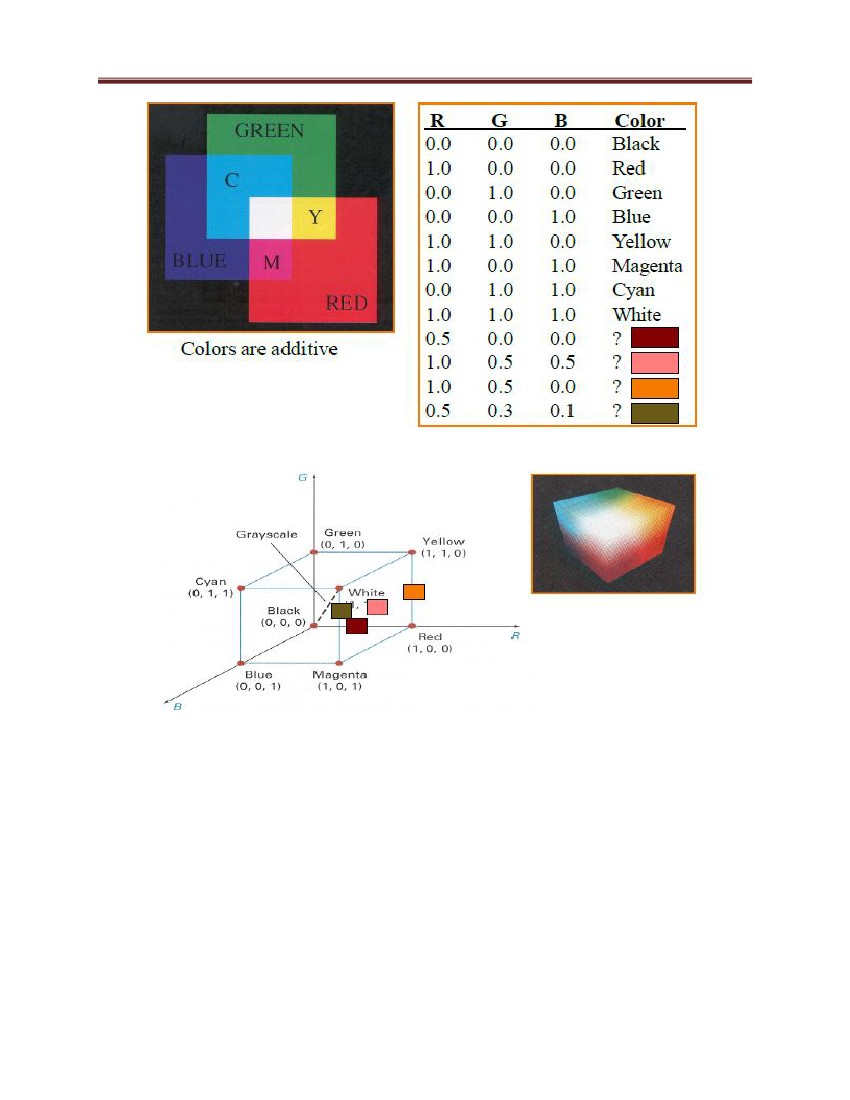


Computer Graphics and Vision

Figure 72: RGB Color Model

Figure 73: RGB Color Cube

123



Computer Graphics and Vision

Figure 74: CMY Color Model

Figure 75: CMY Color Cube

6.7 Linear filter

Linear filters in the time domain process time-varying input signals to produce output signals,

subject to the constraint of linearity.

Most filters implemented in analog electronics, in digital signal processing, or in mechanical

systems are classified as causal, time invariant, and linear. However the general concept of

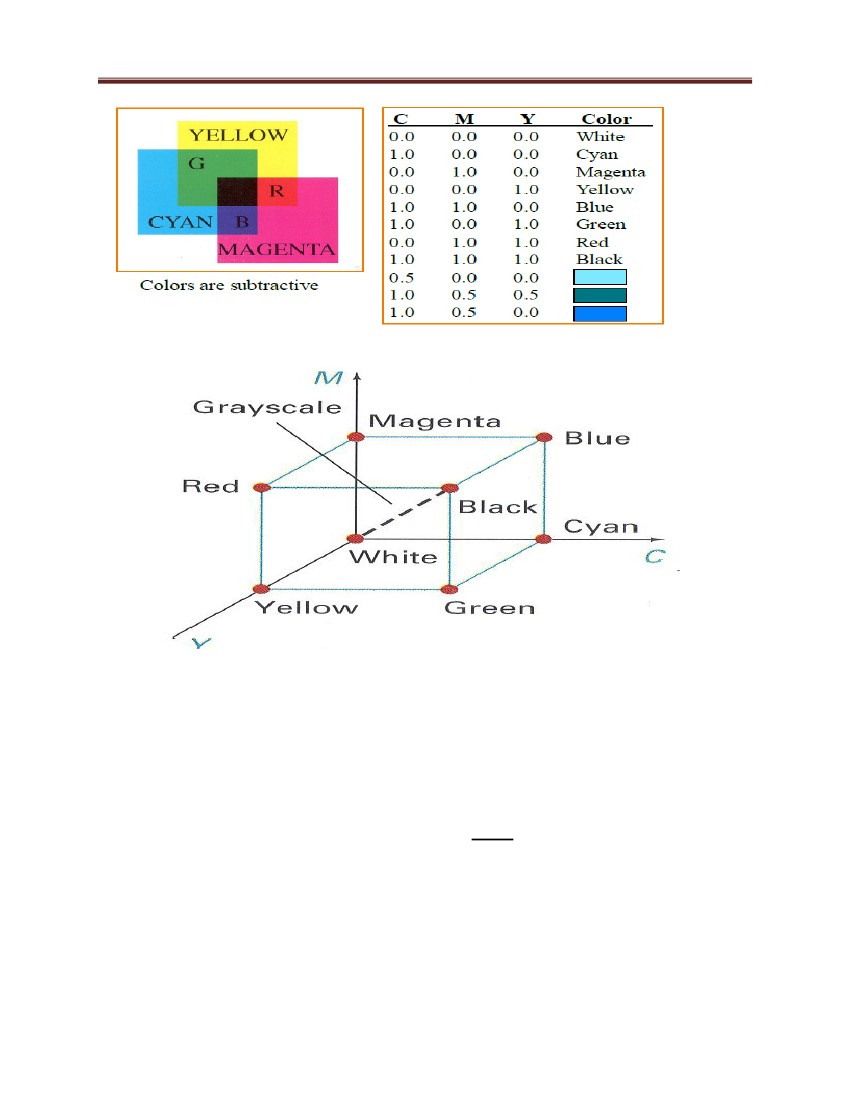
linear filtering is broader, also used in statistics, data analysis, and mechanical engineering

among other fields and technologies. This includes non-causal filters and filters in more than one

dimension such as would be used in image processing; those filters are subject to different

constraints leading to different design methods, which are discussed elsewhere.

124



Computer Graphics and Vision

A linear time-invariant (LTI) filter can be uniquely specified by its impulse response h, and the

output of any filter is mathematically expressed as the convolution of the input with that impulse

response. The frequency response, given by the filter's transfer function

, is an alternative

characterization of the filter. The frequency response may be tailored to, for instance, eliminate

unwanted frequency components from an input signal, or to limit an amplifier to signals within a

particular band of frequencies. There are a number of particularly desirable or useful filter

transfer functions, of which this article will present an overview.

Among the time-domain filters we here consider, there are two general classes of filter transfer

functions that can approximate a desired frequency response. Very different mathematical

treatments apply to the design of filters termed infinite impulse response (IIR) filters,

characteristic of mechanical and analog electronics systems, and finite impulse response (FIR)

filters, which can be implemented by discrete time systems such as computers (then termed

digital signal processing).

6.8 Linear Shift-invariant Systems

Electrical networks or optical imaging systems transform their input (e.g. voltages or light

intensities) as a function of time and/or of space. In general such one- or more-dimensional

transformations S map some input functions f(x,y,z,t) into some output functions g(x,y,z,t):

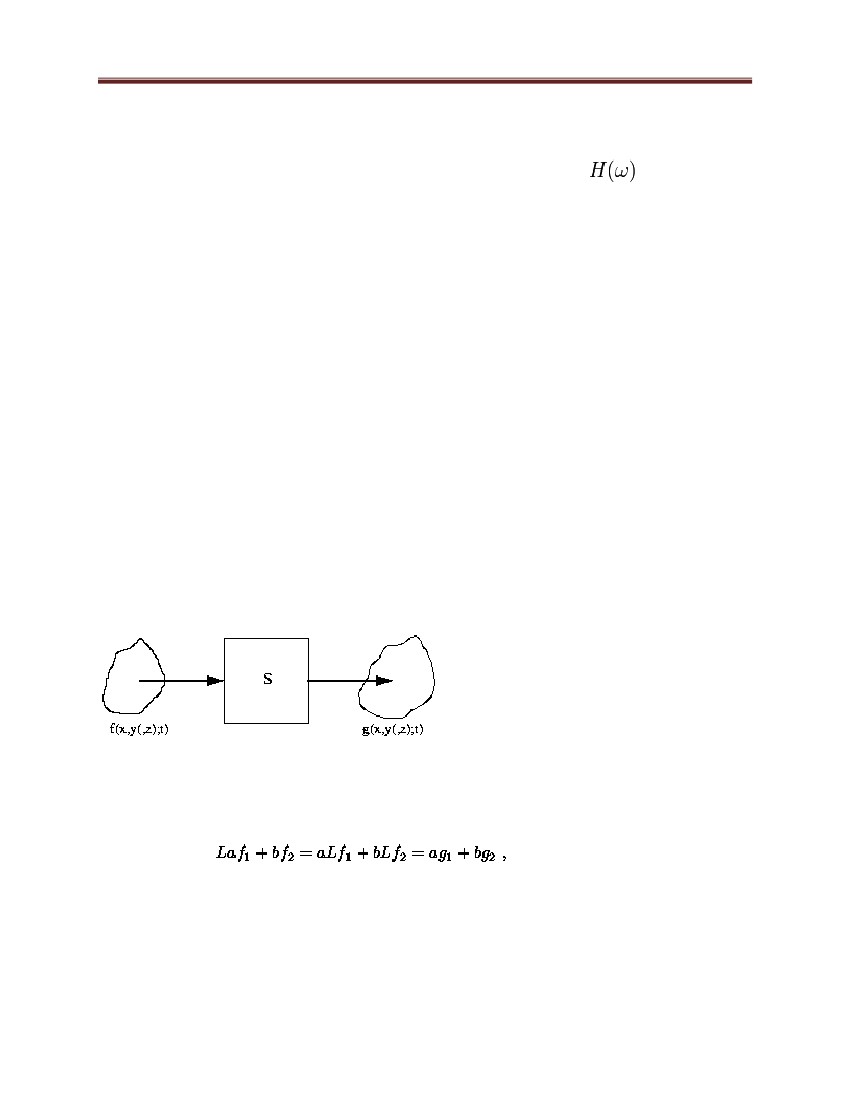
The transformation is called a linear system L, if the following equation holds for all functions f1,

f2 and any a and b:

i.e. an arbitrary function that can be expressed as a sum of several elementary excitations will be

transformed by a linear system as the superposition of the output of these excitations. In general:

125



Computer Graphics and Vision

L is called shift-invariant if and only if a shift (translation) of the input causes the same shift of

the output:

Electrical networks or optical systems are usually treated as time- and space-invariant,

respectively.

To simplify the notation and to derive the computational aspects, we choose a one-dimensional

discrete system. With

we can write the identity:

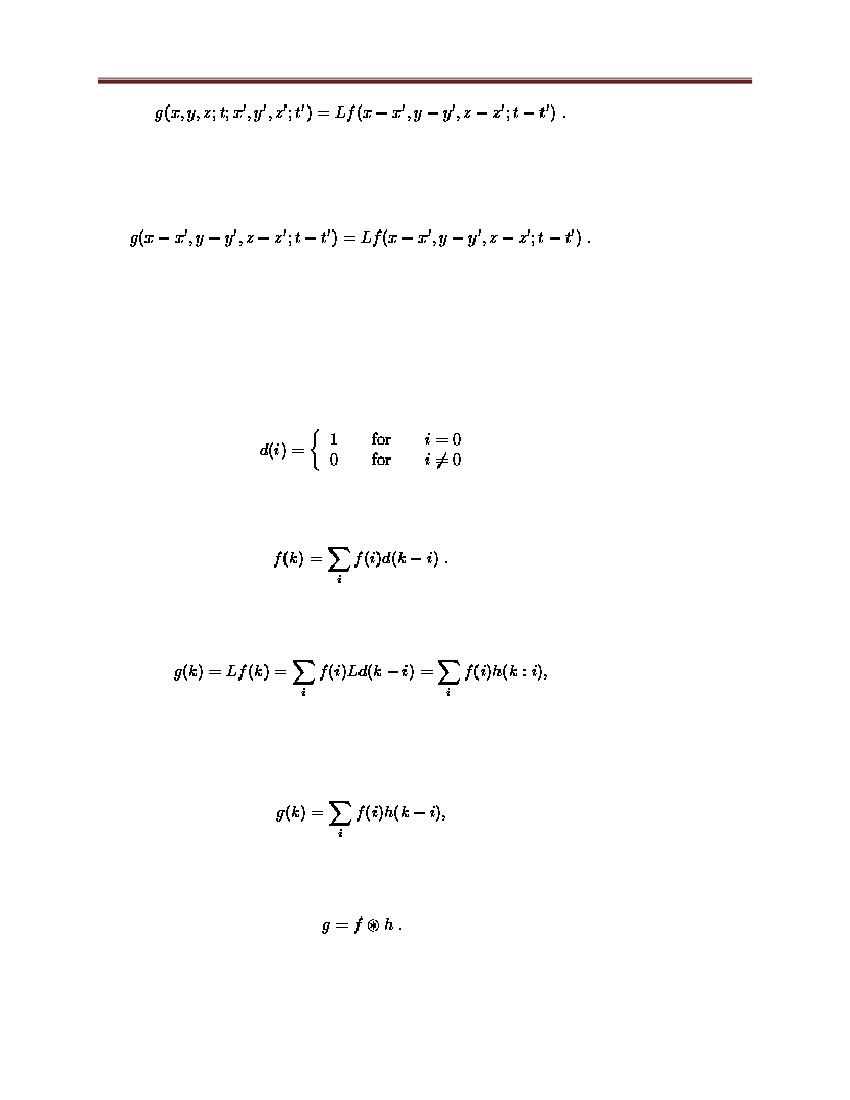
Application of the linear operator L produces:

which is the superposition sum of the shift-varying impulse response h(k;i). If L is shift-invariant,

i.e. h(k-i) = L[d(k-i)], the equation can be written in form of a convolution

or abbreviated:

126



Computer Graphics and Vision

The impulse response h is called the point spread function in the two-dimensional case.

If F,G and H are the Fourier transforms of f,g and h, respectively,

with the frequency response or transfer function H of the linear shift-invariant system L.

6.9 Edge Detection

Edge detection is a technique to locate the edges of objects in the scene. This can be

useful for locating the horizon, the corner of an object, white line following, or for

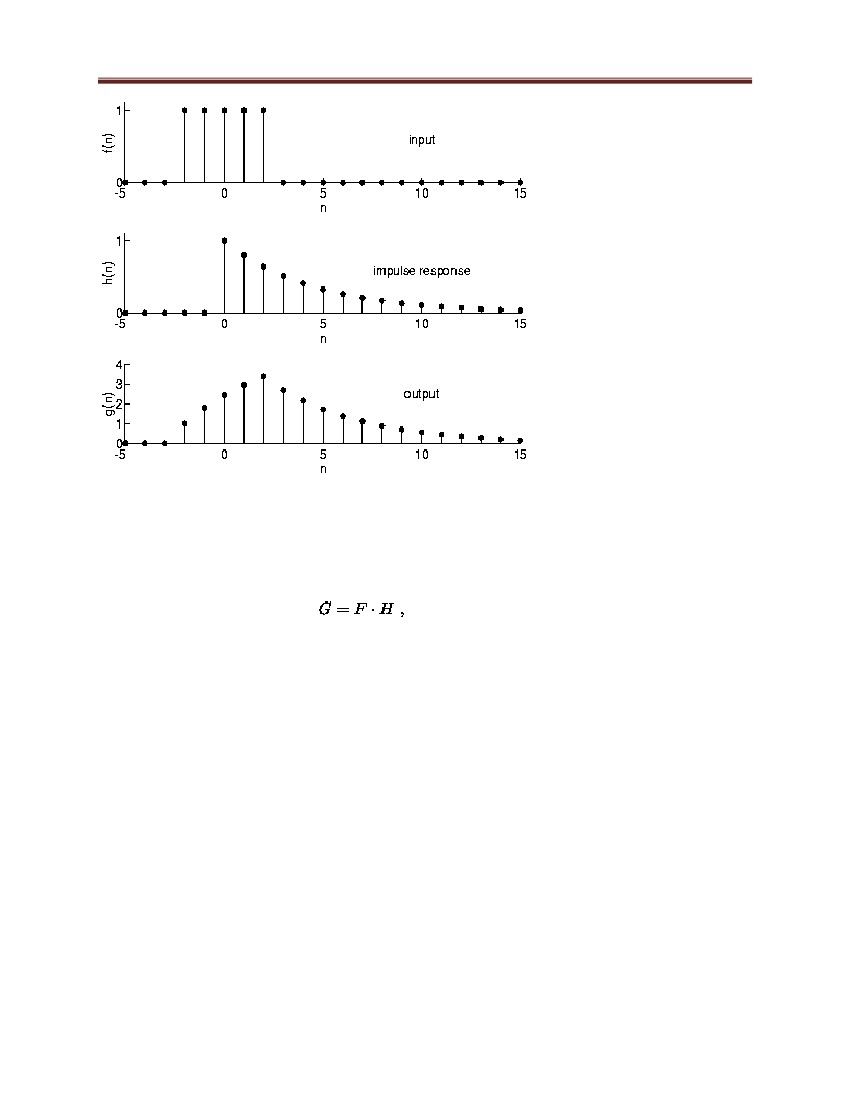
determining the shape of an object. The algorithm is quite simple:

1. sort through the image matrix pixel by pixel

2. for each pixel, analyze each of the 8 pixels surrounding it

3. record the value of the darkest pixel, and the lightest pixel

127



Computer Graphics and Vision

4. if (darkest\_pixel\_value - lightest\_pixel\_value) > threshold)

then rewrite that pixel as 1;

else rewrite that pixel as 0;

What the algorithm does is to detect sudden changes in color or lighting, representing

the edge of an object.

Check out the edges on Mona Lisa:

Figure 76: Edge checking on the picture

A challenge you may have is choosing a good threshold. This left image has a

threshold that is too low, and the right image has a threshold that is too high.

Figure 77: Edge checking on the picture (Cont.)

128

Computer Graphics and Vision

You can also do other neat tricks with images, such as thresholding only a particular

color like red.

6.10. Shape Detection and Pattern Recognition

Shape detection requires preprogramming in a mathematical representation database

of the shapes you wish to detect. For example, suppose you are writing a program that

can distinguish between a triangle, a square, and a circle. This is how it would be

done:

1. run edge detection to find the border line of each shape

2. count the number of continuous edges

3. sharp

change

in

line

direction

signifies

a

different

line

do this by determining the average vector between adjacent pixels

4. if three lines detected, then its a triangle

5. if four lines, then a square

6. if one line, then its a circle

7. by measure angles between lines you can determine more info (rhomboid,

equilateral triangle, etc.)

Figure 78: Recognition between a triangle, a square, and a circle

The basic shapes are very easy, but as you get into more complex shapes (pattern

recognition) you will have to use probability analysis.

129

Computer Graphics and Vision

Also, suppose your algorithm needed to recognize between 10 different fruits (only by

shape) such as an apple, an orange, a pear, a cherry, etc. How would you do it? Well

all are circular, but none perfectly circular. And not all apples look the same, either.

By using probability, you can run an analysis that says 'oh, this fruit fits 90% of the

characteristics of an apple, but only 60% the characteristics of an orange, so its more

likely an apple.' It‘s the computational version of an 'educated guess.' You could also

say 'if this particular feature is present, then it has a 20% higher probability of being

an apple.' The feature could be a stem such as on an apple, fuzziness like on a

coconut, or spikes like on a pineapple, etc. This method is known as feature detection.

Figure 79: Middle Mass and Blob Detection

Blob detection is an algorithm used to determine if a group of connecting pixels are

related to each other. This is useful for identifying separate objects in a scene, or

counting the number of objects in a scene. Blob detection would be useful for

counting people in an airport lobby, or fish passing by a camera. Middle mass would

be useful for a baseball catching robot, or a line following robot.

If there is only one blob in a scene, the middle mass is always located in the center of

an object. But what if there were two or more blobs? This is where it fails, as the

middle mass is no longer located on any object. To solve for this problem, your

algorithm needs to label each blob as separate entities. To do this, run this algorithm:

130



Computer Graphics and Vision

go through each pixel in the array:

if the pixel is a blob color, label it '1'

otherwise label it 0

go to the next pixel

if it is also a blob color

and if it is adjacent to blob 1

label it '1'

else label it '2' (or more)

repeat until all pixels are done

What the algorithm does, it labels each blob by a number, counting up for every new

blob it encounters. Then to find middle mass, you can just find it for each individual

blob.

6.11 Segmentation

• Grouping and Segmentation appear to be one of the early processes in human

vision

• They are a way of \*organizing\* image content into ―semantically related‖

groups

In some applications, segmentation is the crucial step (e.g. some types of aerial

image interpretation.

• Grouping is the process of associating similar image features together

131

Computer Graphics and Vision

Figure 80: Basic ideas of grouping in human vision

• Figure-ground discrimination

– grouping can be seen in terms of allocating some elements to a figure, some to

ground – Can be based on local bottom-up cues or high level recognition

• Gestalt properties

– Psychologists have studies a series of factors that affect whether elements should

be grouped together

• Gestalt properties

– Proximity: tokens that are nearby tend to be grouped.

– Similarity: similar tokens tend to be grouped together.

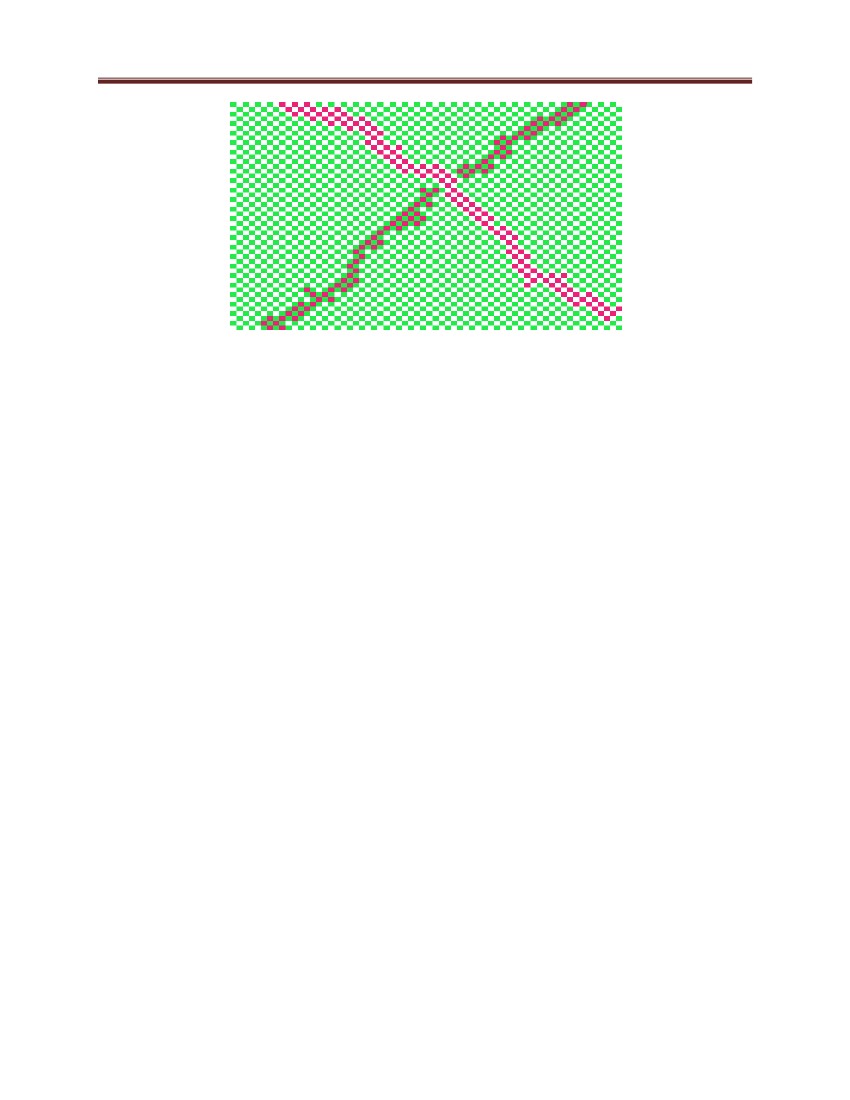
– Common fate: tokens that have coherent motion tend to be grouped together.

– Common region: tokens that lie inside the same closed region tend to be

grouped together.

– Parallelism: parallel curves or tokens tend to be grouped together.

132



Computer Graphics and Vision

– Closure: tokens or curves that tend to lead to closed curves tend to be grouped

together.

– Symmetry: curves that lead to symmetric groups are grouped together.

– Continuity: tokens that lead to ―continuous ‖ (as in ―joining up nicely ‖, rather

than in the formal sense): curves tend to be grouped.

– Familiar Conjuration: tokens that, when grouped, lead to a familiar object,

tend to be grouped together

• Segmentation is the process of dividing an image into regions of ―related

content‖

Figure 81: Segmentation by Clustering

• Cluster

together (pixels, tokens, etc.) that belong together

• Agglomerative clustering

– merge closest clusters

– repeat

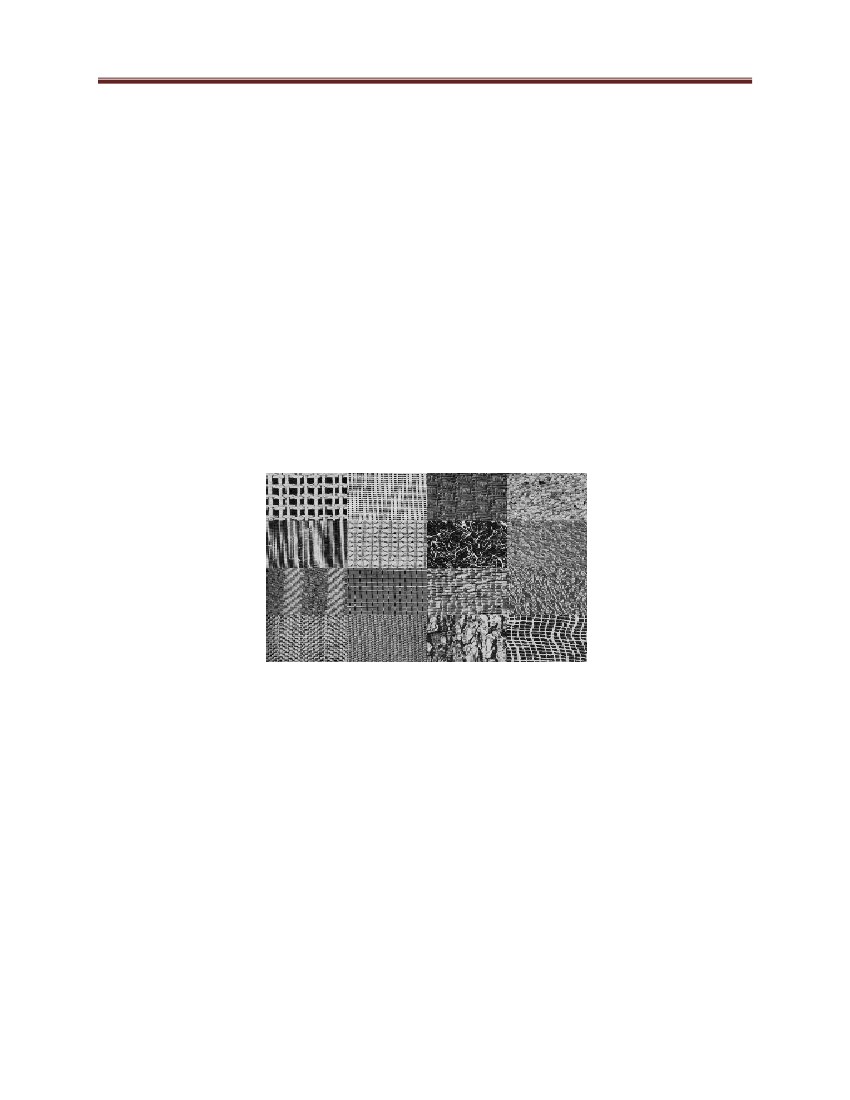
• Divisive clustering

– split cluster along best boundary

– repeat

• Point-Cluster distance

133



Computer Graphics and Vision

– single-link clustering

– complete-link clustering

– group-average clustering

• Dendrograms

– yield a picture of output as clustering process continues

• Data reduction - obtain a compact representation for interesting image data in

terms of a set of components •

Find components that belong together (form

clusters) • Frame differencing - Background Subtraction and Shot Detection

General ideas

• Tokens: – whatever we need to group (pixels, points, surface elements, etc.)

• Top down segmentation: – tokens belong together because they lie on the same

object

• Bottom up segmentation: – tokens belong together because they are locally

coherent. • These two are not mutually exclusive

134



Computer Graphics and Vision

References

1. Computer graphics –C version by Donald Hearn and M. Pauline Baker, second

edition

2. Computer Vision by Dana H. Ballard and Christopher M. Brawn

3. Computer Vision: Algorithms and Applications by Richard Szeliski

4. Dictionary of Computer Vision and Image Processing, Robert Fisher, Ken

Dawson-Howe, Andrew Fitzgibbon, Craig Robertson, Emanuele Trucco,

Wiley, 2005.

5. Computer Vision a Modern approach by David

6. http://luthuli.cs.uiuc.edu/~daf/

7. http://escience.anu.edu.au/lecture/cg/index.en.html

8. http://www.cs.unc.edu/~lazebnik/research/spring08/

A. Forsyth and Jean Ponce

Appendix: Table of Figures

Figure 1: Rendering process ........................................................................................................... 6

Figure 2: Graphics program executes in CPU, filling the frame buffer with values to display..... 8

135



Computer Graphics and Vision

Figure 3: 1-Bit Memory, Monochrome Display (Bitmap Display) ................................................ 9

Figure 4: 3 Bit color Display .......................................................................................................... 9

Figure 5: True color display.......................................................................................................... 10

Figure 6: Los Angeles City by use of CAD process ..................................................................... 11

Figure 7: Example of Entertainment graphics .............................................................................. 12

Figure 8: Example of Educational Graphics ................................................................................. 13

Figure 9: Examples of Visualization Graphics ............................................................................. 14

Figure 10: A Bitmap Image- ......................................................................................................... 19

Figure 11: Vector and Bitmap Differences-vector images are not confined to a rectangular shape.

....................................................................................................................................................... 21

Figure 12 A computer graphics workstation ................................................................................. 22

Figure 13: Basic operation of a CRT ............................................................................................ 23

Figure 14: A raster-scan system displays an object as a set of points across each scan line. ....... 25

Figure 15 Raster scan display. ...................................................................................................... 26

Figure 16: Raster Scan .................................................................................................................. 26

Figure 17: A random-scan system draws the component lines of an object in any order specified.

....................................................................................................................................................... 29

Figure 18: Raster graphics display system.................................................................................... 31

Figure 19: Video Controller coordinate ........................................................................................ 31

Figure 20: Basic video-controller refresh operations.................................................................... 32

Figure 21: A character defined as a rectangular grid of pixel position ......................................... 33

Figure 22: A vector random-scan system .................................................................................... 34

Figure 23: QWERTY Keyboard ................................................................................................... 36

Figure 24: Computer Mouse ......................................................................................................... 36

Figure 25: Track ball ..................................................................................................................... 37

Figure 26: Illustration of the Head-mounted display .................................................................... 38

Figure 27: Wide Screen ................................................................................................................ 39

Figure 28: Square pixels on the display surface ........................................................................... 43

Figure 29: Stair case effect ........................................................................................................... 44

Figure 30: The straight line in Cartesian coordinates ................................................................... 45

136



Computer Graphics and Vision

Figure 31: Basic Line Generation ................................................................................................. 46

Figure 32: Staircase effect ............................................................................................................ 46

Figure 33: DDA line generation ................................................................................................... 48

Figure 34 Slope (m) of line is < 1 (i.e., line 1) ............................................................................. 51

Figure 35: Bresenham Line Generation process .......................................................................... 57

Figure 36: Sampling at x distance ................................................................................................ 58

Figure 37: Property of circle ......................................................................................................... 65

Figure 38: Circle generation (cartesian coordinate system) ......................................................... 66

Figure 39: Circle generation (polar coordinate system) ............................................................... 66

Figure 40: Circle generation (initialization) ................................................................................. 67

Figure 41: Circle generation approaches ...................................................................................... 68

Figure 42: Mid-point between Candidate Pixel at sampling Position xk + 1 along Circular Path 69

Figure 43: Concept of scan line polygon filling ........................................................................... 76

Figure 44: Scan line polygon filling ............................................................................................. 76

Figure 45: Cases for scan line polygon filling .............................................................................. 77

Figure 46: Requirement for image scanning ................................................................................. 79

Figure 47: Scan line polygon filling (a recursive approach)......................................................... 80

Figure 48: Example of the transformation .................................................................................... 81

Figure 49: Illustration of window-to-Viewport ............................................................................ 89

Figure 50: Summary of Transformations ...................................................................................... 90

Figure 51: Wrong visibility illustration ....................................................................................... 92

Figure 52: Correct visibility illustration ....................................................................................... 92

Figure 53: Backface culling .......................................................................................................... 93

Figure 54: Depth sorting illustration ............................................................................................. 97

Figure 55: Relation between computer vision and various other fields ........................................ 98

Figure 56: Digital Imaging.......................................................................................................... 101

Figure 57: Typical hardware components of a computer vision system .................................... 102

Figure 58: A vision-guided welding machine ............................................................................. 103

Figure 59: Face detection ............................................................................................................ 104

Figure 60: Image formation on the backplate of a photographic camera ................................... 105

137



Computer Graphics and Vision

Figure 61: Idealized model that defines perspective projection ................................................ 106

Figure 62: The pinhole imaging models. .................................................................................... 107

Figure 63: The perspective projection equations are derived in this section from the collinearity

of the point P, its image P‘ , and the pinhole O. ......................................................................... 108

Figure 64: Weak-perspective projection illustration................................................................... 109

Figure 65: Orthographic projection. ........................................................................................... 110

Figure 66 Reflection and refraction ............................................................................................ 111

Figure 67: Paraxial refraction ..................................................................................................... 112

Figure 68: A thin lens. Rays passing through the point O are not refracted. Rays parallel to the

optical axis are focused on the focal point F‘ ............................................................................. 114

Figure 69 Left: The main components of the human eye; ---Right: Helmoltz‘s schematic eye. 115

Figure 70: The first photograph on record, la table servie, ........................................................ 116

Figure 71: A CCD Device........................................................................................................... 117

Figure 72: RGB Color Model ..................................................................................................... 123

Figure 73: RGB Color Cube ....................................................................................................... 123

Figure 74: CMY Color Model .................................................................................................... 124

Figure 75: CMY Color Cube ...................................................................................................... 124

Figure 76: Edge checking on the picture .................................................................................... 128

Figure 77: Edge checking on the picture (Cont.) ........................................................................ 128

Figure 78: Recognition between a triangle, a square, and a circle ............................................. 129

Figure 79: Middle Mass and Blob Detection .............................................................................. 130

Figure 80: Basic ideas of grouping in human vision .................................................................. 132

Figure 81: Segmentation by Clustering ...................................................................................... 133

138

