FUNDAMENTALS OF MACHINE VISION

TUTORIAL TEXTS SERIES

- *Fundamentals of Machine Vision*, Harley R. Myler, Vol. TT33
- *Design and Mounting of Prisms and Small Mirrors in Optical Instruments*, Paul R. Yoder, Jr., Vol. TT32
- *Basic Electro-Optics for Electrical Engineers,* Glenn D. Boreman, Vol. TT31
- *Optical Engineering Fundamentals,* Bruce H. Walker, Vol. TT30
- *Introduction to Radiometry,* William L. Wolfe, Vol. TT29
- *An Introduction to Interpretation of Graphic Images,* Sergey Ablameyko, Vol. TT27
- *Thermal Infrared Characterization of Ground Targets and Backgrounds,* Pieter A. Jacobs, Vol. TT26
- *Introduction to Imaging Spectrometers,* William L. Wolfe, Vol. TT25
- *Introduction to Infrared System Design,* William L. Wolfe, Vol. TT24
- *Introduction to Computer-based Imaging Systems,* Divyendu Sinha, Edward R. Dougherty, Vol. TT23
- *Optical Communication Receiver Design,* Stephen B. Alexander, Vol. TT22
- *Mounting Lenses in Optical Instruments,* Paul R. Yoder, Jr., Vol. TT21
- *Optical Design Fundamentals for Infrared Systems,* Max J. Riedl, Vol. TT20
- *An Introduction to Real-Time Imaging,* Edward R. Dougherty, Phillip A. Laplante, Vol. TT19
- *Introduction to Wavefront Sensors,* Joseph M. Geary, Vol. TT18
- *Integration of Lasers and Fiber Optics into Robotic Systems,* Janusz A. Marszalec, Elzbieta A. Marszalec, Vol. TT17
- *An Introduction to Nonlinear Image Processing,* Edward R. Dougherty, Jaakko Astola, Vol. TT16
- *Introduction to Optical Testing,* Joseph M. Geary, Vol. TT15
- *Sensor and Data Fusion Concepts and Applications,* Lawrence A. Klein, Vol. TT14
- *Practical Applications of Infrared Thermal Sensing and Imaging Equipment,* Herbert Kaplan, Vol. TT13
- *Image Formation in Low-Voltage Scanning Electron Microscopy,* L. Reimer, Vol. TT12
- *Diazonaphthoquinone-based Resists,* Ralph Dammel, Vol. TT11
- *Infrared Window and Dome Materials,* Daniel C. Harris, Vol. TT10
- *An Introduction to Morphological Image Processing,* Edward R. Dougherty, Vol. TT9
- *An Introduction to Optics in Computers,* Henri H. Arsenault, Yunlong Sheng, Vol. TT8
- *Digital Image Compression Techniques,* Majid Rabbani, Paul W. Jones, Vol. TT7
- *Aberration Theory Made Simple,* Virendra N. Mahajan, Vol. TT6
- *Single-Frequency Semiconductor Lasers,* Jens Buus, Vol. TT5
- *An Introduction to Biological and Artificial Neural Networks for Pattern Recognition,* Steven K. Rogers, Matthew Kabrisky, Vol. TT4
- *Laser Beam Propagation in the Atmosphere,* Hugo Weichel, Vol. TT3
- *Infrared Fiber Optics,* Paul Klocek, George H. Sigel, Jr., Vol. TT2
- *Spectrally Selective Surfaces for Heating and Cooling Applications,* C. G. Granqvist, Vol. TT1

FUNDAMENTALS OF MACHINE VISION

Harley R. Myler
University of Central Florida

Tutorial Texts in Optical Engineering
Volume TT33

Donald C. O'Shea, Series Editor
Georgia Institute of Technology

SPIE Optical Engineering Press
A Publication of SPIE—The International Society for Optical Engineering
Bellingham, Washington USA

Library of Congress Cataloging-in-Publication Data

Myler, Harley R., 1953–
Fundamentals of machine vision / Harley R. Myler.
p. cm. – (Tutorial texts in optical engineering; v. TT33)
Includes bibliographical references and index.
ISBN 0-8194-3049-8
1. Computer vision. I. Title. II. Series.
TA1634.M954 1998
006.3'7—dc21 98-31172
CIP

Published by

SPIE—The International Society for Optical Engineering
P.O. Box 10
Bellingham, Washington 98227-0010
Phone: 360/676-3290
Fax: 360/647-1445
Email: spie@spie.org
WWW: http://www.spie.org/

Printed in the United States of America.

PDF ISBN: 9781510608023

To my mother, Anna,
my wife, Nancy, and
my daughter, Krifka

SERIES INTRODUCTION

The Tutorial Texts series was begun in response to requests for copies of SPIE short course notes by those who were not able to attend a course. By policy the notes are the property of the instructors and are not available for sale. Since short course notes are intended only to guide the discussion, supplement the presentation, and relieve the lecturer of generating complicated graphics on the spot, they cannot substitute for a text. As one who has evaluated many sets of course notes for possible use in this series, I have found that material unsupported by the lecture is not very useful. The notes provide more frustration than illumination.

What the Tutorial Texts series does is to fill in the gaps, establish the continuity, and clarify the arguments that can only be glimpsed in the notes. When topics are evaluated for this series, the paramount concern in determining whether to proceed with the project is whether it effectively addresses the basic concepts of the topic. Each manuscript is reviewed at the initial state when the material is in the form of notes and then later at the final draft. Always, the text is evaluated to ensure that it presents sufficient theory to build a basic understanding and then uses this understanding to give the reader a practical working knowledge of the topic. References are included as an essential part of each text for the reader requiring more in-depth study.

One advantage of the Tutorial Texts series is our ability to cover new fields as they are developing. In fields such as sensor fusion, morphological image processing, and digital compression techniques, the textbooks on these topics were limited or unavailable. Since 1989 the Tutorial Texts have provided an introduction to those seeking to understand these and other equally exciting technologies. We have expanded the series beyond topics covered by the short course program to encompass contributions from experts in their field who can write with authority and clarity at an introductory level. The emphasis is always on the tutorial nature of the text. It is my hope that over the next few years there will be as many additional titles with the quality and breadth of the first ten years.

Donald C. O'Shea
Georgia Institute of Technology

CONTENTS

PREFACE

Machine vision has many definitions and is called by various names, often depending on the discipline that one practices. *Computer vision*, *image understanding*, *scene analysis*, and *robot vision* are some of the terms encountered. These terms are derived from computer science, signal processing, pattern recognition, and robotics studies, respectively. In this text, machine vision is explicitly defined as the study and implementation of systems that allow machines to recognize objects from acquired image data and perform useful tasks from that recognition. The term *systems* in this definition includes both hardware and software, with the restrictions that the hardware is constrained to acquisition and processing equipment and that the algorithms perform recognition and reasoning only. This eliminates issues of robots or vehicles, which are best left to other studies. The concept of a *useful task* helps confine the definition to a manageable level, yet be expandable to include large-scale general purpose vision systems.

Although intended as a supplement to a classroom short-course, the book stands alone as a useful self-study guide and is certainly usable as a primary text or supplement for a more extensive course offering. The organization is novel in that machine vision has essentially come of age and we may now view it (no pun intended) as an established and respected field of research and application with a strong theoretical foundation. Without foundation we simply cannot predict outcomes. Fortunately, it is now possible to engage in machine vision design with high probability of success. Because of this, the text takes the reader from fundamentals drawn from image processing and computer graphics to the methods of applied machine vision techniques. This background is then applied to the largely theoretical basis of human vision, which is, of course, the ultimate measure against which a machine vision system is compared.

There is a tendency for writers to feel that "more is better" and to burden a book with excessive coverage to the extent that the reader is overwhelmed. Since the literature on machine vision is so extensive, including a large number of texts, I have restricted the content to include only what is necessary to allow the reader to understand and construct machine vision systems that perform, once again, useful tasks based on the current state of the art. Additionally, I have prepared an Annotated Bibliography as an assist in navigating a sampling of this vast literature.

To fail to include open-ended discussion would expose the book to speedy decline with respect to a rapidly evolving area, so I have endeavored to leave some discussion without closure so as to stimulate new and creative work in this fascinating and dynamic field. As life-long learners we must strive to advance and grow, and it is my sincere hope that this book assists you in that process.

H. R. Myler
August 1998

CHAPTER 1
VISION IN HUMANS AND MACHINES

In order to address the development of advanced vision capability in the machine, it is useful to begin with an examination of the visual systems of humans and animals. The complexity of these systems illustrates what is possible, and in some instances has shown us how we can construct machine vision systems with the efficiency and capabilities that nature has employed. There are, however, limitations on natural systems that can be overcome with machines, as there are capabilities that exist in biological systems that we have yet to duplicate in the artificial.

1.1 VISUAL SYSTEM MECHANICS

The human visual system (HVS) begins with the eye and ends in the mind, but where exactly is the mind? This question must be left to further study; however, it is possible to trace the neural pathways from the eye to the brain. The human eye is an extension of our nervous systems unlike many lower animals where the eye(s) evolved from the skin. Because of this, the human eye performs a substantial amount of preprocessing before the final signal arrives at the brain. Modern electronic cameras are now being produced with processing capabilities for image filtering, data reduction, and enhancement, much in the same way that the biological eye preprocesses image data.

The human eye is a complex organ of signal detection that supplies the brain with a high-resolution color image. A simple diagram of the human eye is shown in Figure 1-1. Light enters the eye through the cornea, a clear, dome-shaped lens. Here the light undergoes an initial refraction before it reaches the pupil, which acts as a variable aperture to control the amount of light entering the interior of the organ. The focusing lens is a bean-shaped bundle of transparent protein fibers. Muscles surrounding the lens distort it and thus control the focus of light onto the light sensitive nerve layer, the retina, at the back inner surface of the eye. The retina consists of approximately 100 million sensors whose outputs are summed and thresholded by cell junctions called ganglia. The output of the ganglia, the nerve fibers, pass over the surface of the retina and may be seen by using an ophthalmoscope, a special monocular viewer designed to allow easy examination of the interior of the eye. The nerve

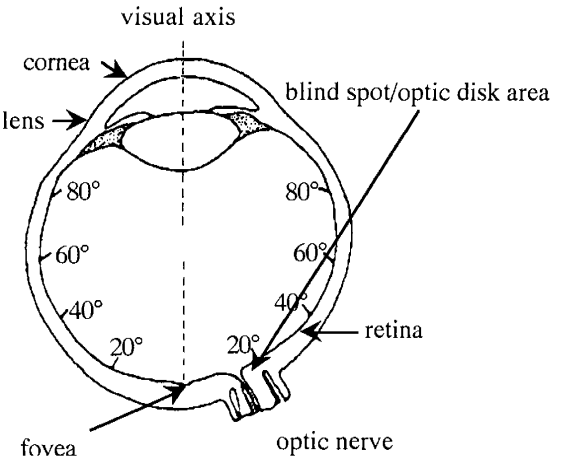


Figure 1-1. Cross-section schematic of the human eye.

fibers appear as wispy blue strands that meet to form the optic nerve, the output channel of visual data leaving the eye. The optic nerve consists of approximately 800,000 nerve fibers and so the reduction from number of light-sensing cells to transmission cells is roughly ten to one. This factor is important to visual perception and is discussed later. At the point where the nerve fibers leave the back of the eye to form the optic nerve there are no light sensors. This causes the so-called "blind spot," and this region of the retina is known clinically as the "optic disk." You can experience the lack of perception at the blind spot by closing your right eye and focusing on the plus symbol shown in the graphic below:

As you vary the distance of your left eye to the page, at some point the star will disappear. At that distance the star is being imaged onto the blind spot of your left eye. One question that has been raised asks why we do not have holes in our visual field. The answer is that we do, but the brain compensates for the lack of sensors and fills in the images received from the eyes to give us a homogeneous panoramic view.

The light-sensing cells fall into one of two classes: cones, for color or photopic vision, and rods, for monochromatic or scotopic vision. These cells are directed towards the back of the eye because the evolution of the organ caused an inversion to take place such that the nerves travel across the surface

of the retina to the exit point at the optic disk. The blind spot would not occur if the light-sensing cells did not have this orientation.

The distribution of sensors at the back surface of the eye is shown in Figure 1-2. As the optical axis is approached from either side, the concentration of cones increases and the number of rods decreases to zero. The peak of the cone distribution is the fovea, as indicated on the eye schematic (Figure 1-1). On the right side of the distribution diagram, the nasal side, at approximately 15° is a region with no sensors at all. This is the blind spot. The sensor distribution diagram shown in Figure 1-2 represents a cross-section taken horizontally and through the center of both the fovea and blind spot. These elements of eye structure are important in that the neural processing eliminates any missing area in our visual field caused by the blind spot. This distribution of sensor density prompted the use of dynamic variable resolution in high-speed aircraft simulation display systems. In these systems, the pilot is fitted with an eye-tracking device that informs the graphics-generating computer where the pilot is looking. The computer then generates a highly detailed image in this region and a less detailed image elsewhere, thus conserving computer resources.

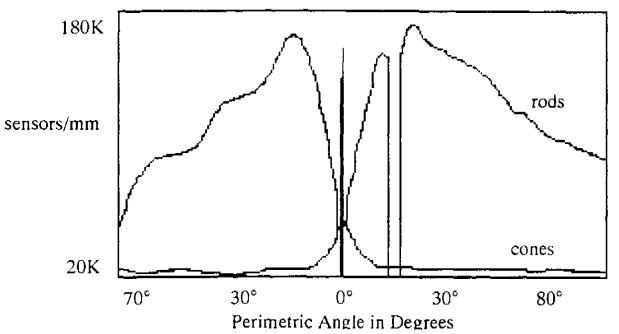


Figure 1-2. Sensor distribution in human retina.

After leaving the back of the eyes, each of the optic nerves split near the base of the brain and rejoin at a nerve center called the optic chiasm. This splitting gives rise, in part, to the right-brain, left-brain phenomena, where half of the image of each eye is interpreted by the left half of the brain and half by the right. Specifically why this occurs and what ultimate role it plays in visual perception is not completely known. For the interested reader, a thorough treatment of the structure and physiology of the human visual system may be found in Zeki's monograph.[1]

In addition to the complexity that arises from the multiple neural pathways from the eye to the brain, another complexity is introduced because the eye is in constant motion. This occurs because the light- sensing cells are differential

in action and therefore react only to changes in light intensity. To form an image, the light must be scanned across the retinal surface. This function is performed by a set of six muscles that attach to each eye and are anchored to the interior surface of the eye socket in the skull. The eye floats in the socket surrounded by soft tissues and gimbaled by the muscles. The small involuntary eye movements that serve to fine position the eye and to track objects are called saccades. Microsaccades are even smaller movements that produce the necessary motion to refresh the image. This is discussed further in Section 1.4, Motion Perception.

1.2 Visual Perception

The image formed on the retina has persistence within the HVS on the order of milliseconds. This delay is due to the slow reaction of the chemical-based sensors (cones and rods) and in the time it takes data to be transmitted to the visual cortex in the brain. This delay establishes the basis of our perception of moving pictures produced by films or television. The rate at which we establish continuity between frames of image sequences is known as the critical fusion frequency or alternately as the flicker fusion rate. For humans, the actual rate varies among individuals and is coupled to visual acuity and other factors. Generally, 24 frames per second are required to achieve image fusion and remove flicker. One often hears the value of 30 frames per second (fps) as the rate for "real-time" video. This is an artifact of technology where the 30-fps refresh rate of television was established from the alternating current frequency of 60 Hz in the United States. This rate is more than adequate to allow humans even with hyperacuity to achieve fusion of video images. The fact that our visual systems process a sequence of images means that a degree of noise filtering is possible. The frame-to-frame sequence is averaged by the system, allowing any noise of duration less than the flicker fusion rate to be averaged out. The phenomenon is easy to observe by using the frame pause or slow motion effect on a VCR. The still image of a single frame appears grainy without the picture clarity seen when the tape is played at normal speed. This, of course, also establishes an upper limit on the detection of a fast-moving object.

Surprisingly, few objects in nature are undetectable to our visual system because of system response time. However, we cannot see many human-fabricated objects that move at high speeds. The in-flight trajectories of bullets fired from weapons, for example, are invisible to us. Nevertheless, these flights are not beyond the detection and tracking ability of our machines. This is an important factor to consider in the design of machine vision systems. The machine can see far more than we humans can. There is a tendency to try to model the machine's perception based on our own perception—and this is certainly an important and valid area of research; however, keep in mind that there are things a machine can detect and process that we cannot. Machine

vision can be approached as a study in giving the machine the perceptual powers of human vision (along with the limitations), or as a method of allowing the machine to see only what it needs to see (as in factory automation). The problem of emulating human perception in the machine is far from being solved in any general sense, but machine vision systems designed to operate in well-defined domains are often solvable by well-chosen hardware, work cell environments, and software.

In HVS terms, we can define brightness as the degree of light stimulus. Brightness is perceived by the HVS as a logarithmic function of light intensity incident on the eye. Contrast is defined as the difference in light intensity between perceived objects. Contrast threshold is a just noticeable difference that is dependent on previous (or current) stimulus. This dependence is termed *adaptation*. The adaptation threshold affects the detection power of the eye and is also a function of sensor density; for example, the threshold is lower at the periphery of the retina.

In Figure 1-3 the subjective brightness is plotted as a function of intensity. A simple example of adaptation effects occurs when one views automobile headlights at night and during the day. At night, the eyes are dark adapted and headlights appear close to the glare limit. During the day, when we are light adapted, headlights left on in a parking lot, for example, are perceived as dim because the brightness adaptation range is low on the actual intensity curve.

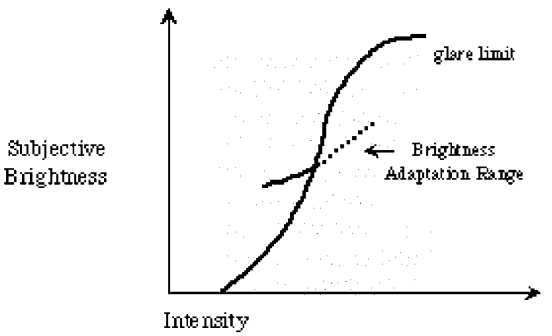


Figure 1-3. Brightness adaptation range.

At very low intensity, we cannot detect an object at all, and at very high intensity detection is restricted by the glare limit. The upper and lower limits will vary between individuals. On the side of the main curve is a secondary curve, the brightness adaptation range. This range (the short crossing line) moves up and down the main curve as our eyes adapt to the ambient light intensity. The dotted portion of the adaptation curve is such because we cannot actually detect intensity beyond the main curve, but it is shown to illustrate the variation of subjective brightness. The slope of the brightness adaptation range is not as steep as the actual brightness detected. A number of explanations have

been given for why these effects occur, and their discussion is beyond the scope of this book; however, we examine two approaches to the explanation, spatial frequency filtering and neural network processing, later in this section.

Perceptual psychologists, using diagrams similar to those shown in Figure 1-4, measure brightness adaptation and contrast effects. The intensity of the center circle with respect to the rectangular background is varied, and the images are evaluated across a population of human subjects. Under varying lighting conditions, the subjective contrast will vary from individual to individual. For example, an individual may not be able to detect that a circle is present in the center graphic while another does. This is because human visual perception varies across the population and is affected by age, eye color, and race, whereas the machine has no such restrictions. A machine would detect the existence of the circles as long as a differential in brightness is detectable by the sensor employed.

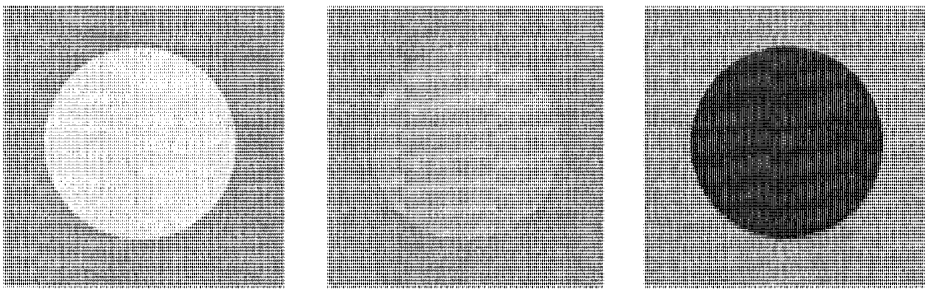


Figure 1-4. Evaluation diagrams for contrast effects.

Adaptations that occur after perceptions of a series of flash illuminations are called metacontrast effects. The contrast threshold *rises* before the perception of a subsequent flash. This effect is best noticed with cameras that feature "red eye reduction" flash mode. The camera generates a fast sequence of flashes to constrict the pupil and thus reduce the likelihood of a red reflection from the retina. This staged constriction of the pupil gives rise to metacontrast effects.

If background illumination is uniform and extensive, a contrast threshold AB is approximately proportional to an illumination B over a wide range. This is known as Weber's Law. Because brightness adaptation is simply a change in overall brightness sensitivity, similarly contrast sensitivity is a measure of brightness level discrimination. Using evaluation diagrams like those depicted in Figure 1-4, it is possible to show Weber's ratio graphically. Figure 1-5 shows two plots in the right column that were determined by varying brightness values in the test images shown in the left column. In both cases, the

value ΔB/B yields the Weber ratio, which is approximately 20%. The upper plot shows the Weber ratio as background brightness B is varied.

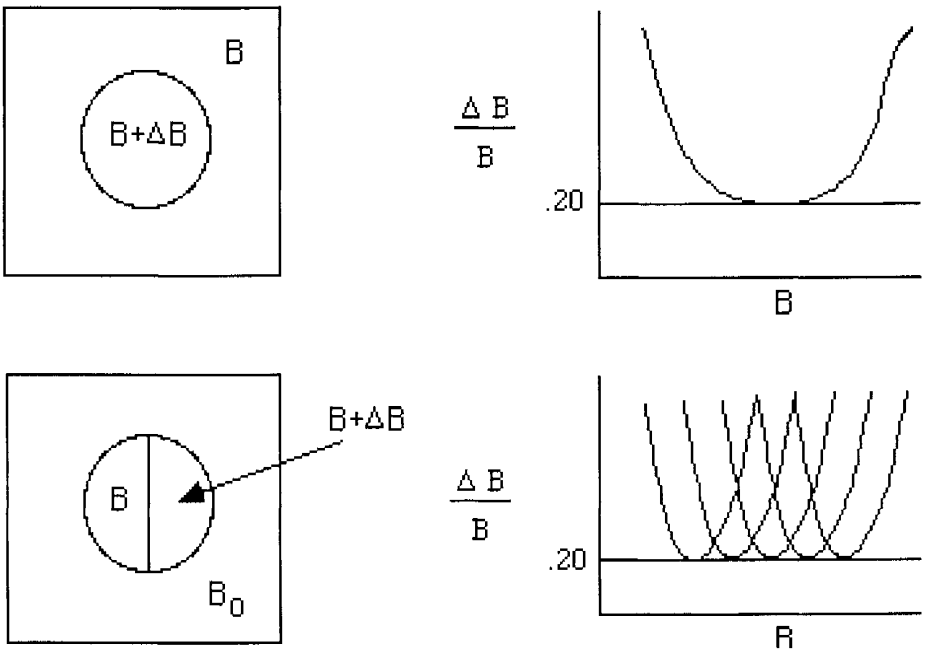


Figure 1-5. Weber ratio, brightness constancy.

The second plot shows what happens for varying values of B_0. This family of curves, each representing a different background B_0 intensity, illustrates what is called brightness constancy, or when object and background retain constant contrast, and an object varies in intensity but *appears* as a constant intensity.

These contrast effects can be attributed, in part, to the logarithmic response of the HVS. This is understandable because of the inherent time delays of the system and the integrations that occur as a function of biological sensor response. Consider two images of circular patterns, Figure 1-6(a) and 1-6(b), and the images that result when the log of brightness values is taken, 1-6(c) and 1-6(d). The (a) image center is substantially lighter than (b). In the log result, the difference is less pronounced, which is a form of brightness constancy. This may be somewhat easier to see in Figure 1-7, where the cross-section profiles of 1-6 (a-d) have been plotted.

The brightness contrast processing that takes place in the HVS may be used in machine vision processing to enhance images prior to higher level object processing. The application of a log function to an image will perform an enhancement in some applications and can be used to compensate for poor

lighting. Brightness adaptation techniques have been employed to reduce the "blooming" effect that can occur in night vision systems that have been overcome by an intense light source, as in the case of a flare. Essentially, these systems perform an automatic gain control on the image data acquired.

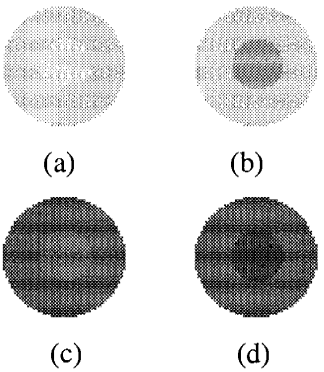


Figure 1-6. Logarithmic response.

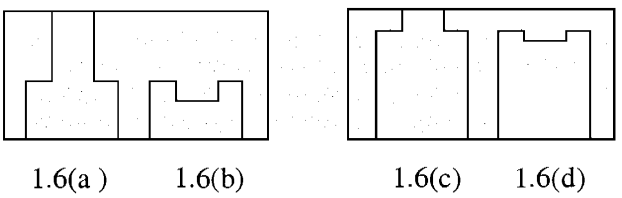


Figure 1-7. Logarithmic response plots.

Resolution of the HVS can be described by a *Modulation Transfer Function* (MTF) that describes an optical system as a bandpass filter. This shape can be observed in sinusoidally modulated patterns such as that shown in Figure 1-8. Note how the density of the lines increases from the left of the figure to the right. The lines-per-area of this figure is representative of how we define *spatial frequency*. If you examine the region of highest spatial frequency (greatest density of lines) using your peripheral vision, you will notice a blurring that takes on a distinctive shape. This blurring is a result of your visual system reaching its capacity with respect to resolving, i.e., distinguishing, the high spatial frequencies of the closely spaced lines. This defines the spatial frequency response of the HVS at the periphery of your vision, where the number of image sensors in the retina is lower (see Figure 1-2). When the gaze is directed at the close-spaced lines they are resolved because of the greater density of sensors at the focal point (the *fovea*) at the back of the retina.

Figure 1-8 also shows moiré patterns, which are a function of spatial aliasing. Here you see swirled line patterns as the spacing of the sinusoidal lines gets closer as the spatial frequency increases. This is not an illusion, but a consequence of trying to print an image requiring greater resolution than the output media can provide. In this case, the media is a 600 dpi laser printer. The point here is that either the acquisition or display mechanism must have enough resolution to accommodate the smallest line in the image.

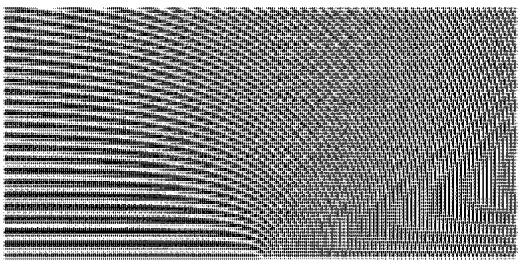


Figure 1-8. Sinusoidally-modulated pattern.

Figure 1-9 shows a column of black dots with horizontal lines extending from the dot edges. To the right of this column are three columns of shaded dots that represent large, medium, and small sensors. The large sensors are the size of two of the black dots, the medium sensors are the same size as the dots, and the small sensors are one-half the size of the dots. What the sensors *detect* is shown to the right of the arrow in the figure, where the column of black dots has been overlayed onto the sensors. The large sensors will not be able to *resolve* the size of the dots, the medium and small sensors, however, will be able to. The medium sensors are at twice the spatial frequency of the dots—those readers with a signal processing background will recognize this as the *Nyquist rate*. Selection of sensors is critical to machine vision applications, because the number of sensors is directly proportional to cost. Too few sensors will not do the job, too many will be overkill and add unnecessary cost to the system.

Now consider Figure 1-10. Assume that we are looking at a graphic similar to one of the squares in Figure 1-4 where a light circle appears on a dark background. If we polled a row of sensors on the retina that were located across the center of the figure as it was imaged by the eye (ignoring motion and other effects) and then plotted the light intensity at those sensors as a function of position, we would have a plot similar to Figure 1-10(a). Now apply a signal processing technique, such as the Fourier transform, that determines the frequency components of the curve. This we plot as Figure 1-10(b), where the x-axis is now frequency of sinusoidal components and the y-axis is their magnitudes. Since this is an imaging activity, one can label the frequency axis as *spatial* frequency with no loss of generality. This transition is

possible because our data originally represented one dimension of a lines-per-area, or image, data set. In Section 2.3 we examine the Fourier transform in more detail as it applies to images, but for now it is sufficient to use a one-dimensional transform and treat the data as a single-dimensional signal for purposes of discussion.

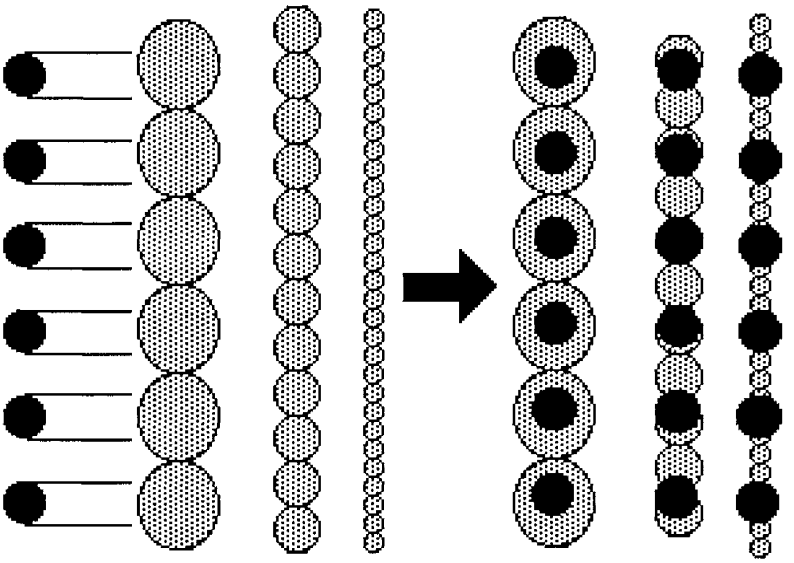


Figure 1-9. Resolution of sensors.

Those familiar with signal analysis will recognize Figure 1-10(b) as a crude attempt at depicting the positive values of a sinc function drawn by hand. Now consider Figure 1-10(c), the MTF, or modulation transfer function. This curve represents the proportion of spatial frequencies that the visual system is allowing to pass. At the low end there is truncation as well as at the high end—a bandpass filter.

Now apply the filter, Figure 1-10(c), to the frequency spectrum, Figure 1-10(b). The result will be a new spatial frequency spectrum where only what is below the MTF filter line is permitted to remain, as shown in Figure 1-10(d). If we then perform an inverse operation on just the magnitude data below the filter line, the inverse Fourier transform, we will return to the "position" domain as before, but our result will look as shown in Figure 1-10(e). Note that the sharp corners that we had before are now rounded, but peaked both in a negative direction as at the base of the rectangle and positively at the top. In essence, the filtering has *enhanced* the edges of the figure. In other words, the edges are brighter with respect to the rest of the image.

One explanation of this phenomenon is found in neurophysiology. Earlier it was shown that the number of actual light sensors in the retina outnumbered

the channels within the optic nerve carrying the image signals to the brain and that the detectors were grouped by ganglia cells. This reduction in data has a low pass filtering effect on the data, giving rise to the low-end attenuation of the MTF, and the corresponding rounding and depression in the output curve. On the high end, there is a general property of nerve cells known as *lateral inhibition*. When two adjacent cells receive stimulation, they will laterally inhibit the output of their neighbors proportional to the input that they themselves are receiving. If two cells receive the same amount of stimulation, the effect of their inhibitory behavior on each other will effectively cancel. However, when one cell receives more stimulation than another, the effective output of the cell receiving less stimulation will be suppressed!

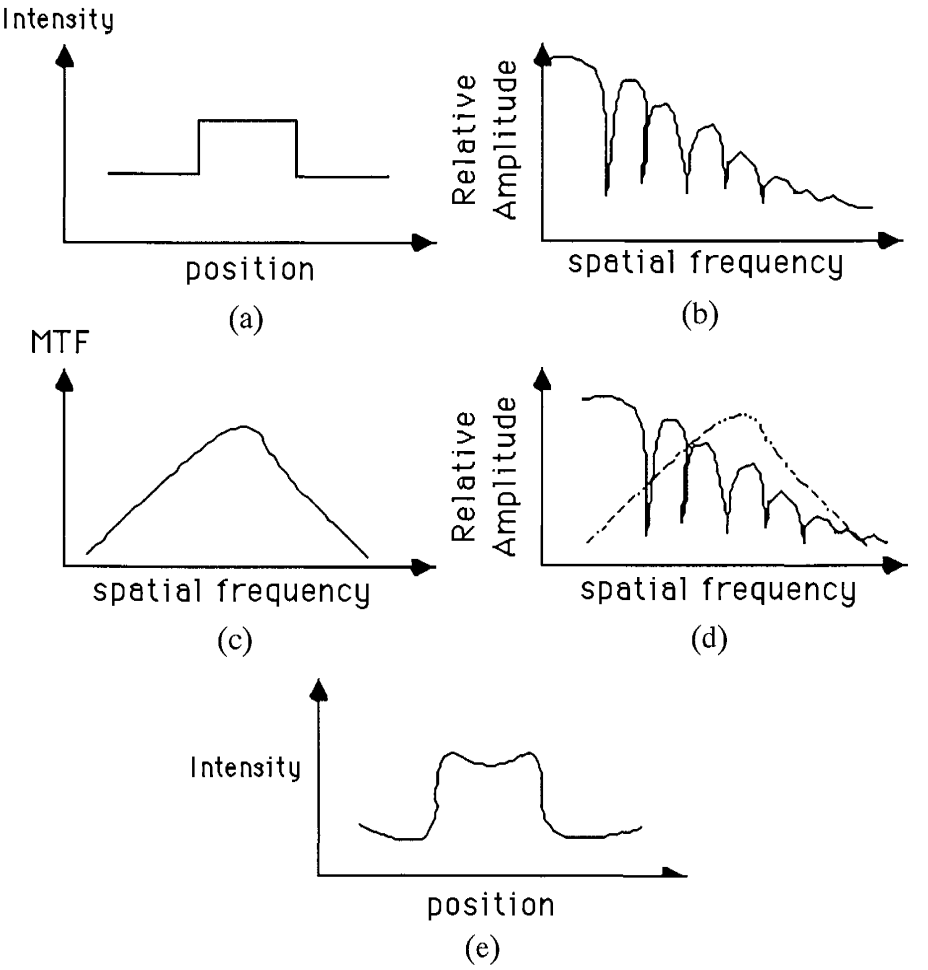


Figure 1-10. Filtering effect of HVS.

To see this more clearly, consider Figure 1-11. Two adjacent cells are represented by circles, and the relative strengths of their input and output signals are indicated by the arrowhead lines. On the left, equal input, equal inhibition, equal output. On the right, unequal input. Now the cell on the left not only inhibits more, but also the cell on the right inhibits less, causing a substantial variance in outputs compared to inputs. With a row of cells, such as that shown in Figure 1-12, the effect will be roughly equivalent to the effect shown in Figure 1-10(e), edge enhancement brought on by MTF filtering. The final visualization is, of course, when there is a two-dimensional array of cells. Now the effect will enhance edges in all directions. One can consider this a biological method of edge detection. The importance of edge enhancement and detection will show up later when we discuss the evaluation and implementation of algorithms to achieve this by electronic imagery.

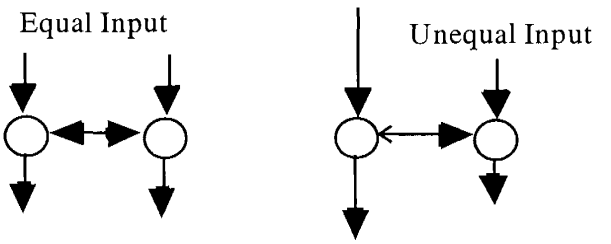


Figure 1-11. Two cell effects of lateral inhibition.

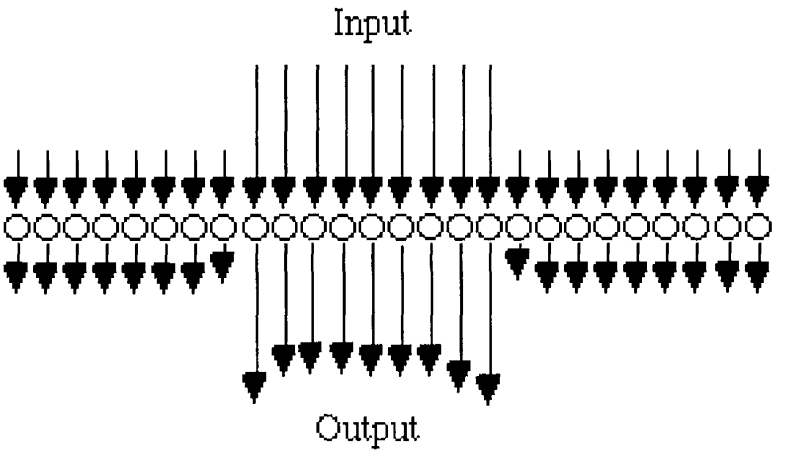
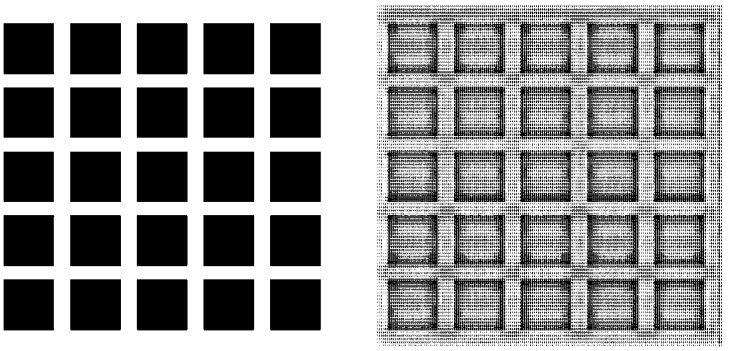


Figure 1-12. Lateral inhibition on a linear array of cells.

Spatial filtering in the HVS gives rise to a number of optical illusions—one of the most famous being the Hermann Grid Illusion, shown here in

Figure 1-13. The actual illusion is shown in Figure 1-13(a) and a bandpass-filtered version using the MTF function from Figure 1-10(c) is shown in Figure 1-13(b). The illusion is the appearance of ghostly gray squares at the white crossways of the grid (you get the same illusion with white squares on black). The filtered version shows that the ghosts are really there. But why do they disappear when you look directly at them?



(a) (b)

Figure 1-13. Hermann grid illusion, (a) the illusion, (b) after bandpass filtering.

The "ghostly" squares are a result of the sparse number of sensors that are present at the periphery of your vision. Recall that the density of sensors falls as we move away from the fovea (see Figure 1-2). Think of this in terms of information. As the number of possible values of representation is reduced, the information capacity is correspondingly reduced. Since the squares represent sequences of rectangular spatial frequency, the lower sampling ability at the periphery of the retina allows artifacts to appear between the squares. These artifacts are due to the lack of high spatial frequency sensing needed to produce the sharp corners of the squares. This is the filtering effect discussed earlier. When you look directly at the space between squares, the high sensor density of the fovea resolves the high spatial frequencies and the ghostly artifact disappears.

The higher the resolution that an image has, the more information it conveys. Figure 1-14 below shows two examples of the same image of a collection of office tools against the cooling vents of a display monitor. The first image [1-14(a)] has twice the resolution as the second [1-14(b)]. We say that the first image has been *sampled* at a higher spatial frequency than the second. Notice the loss in spatial detail between the images; this is a result of the *subsampling* that has occurred in the second image. Subsampling refers to less than desirable sampling, although in some applications subsampling is

used to reduce the amount of data processed. Also note that the images are the same size, indicating that the sensors used to produce the first image are smaller (and more numerous) than those used to produce the second image. In Chapter 2, Image Processing, we explore the issue of sampling further.

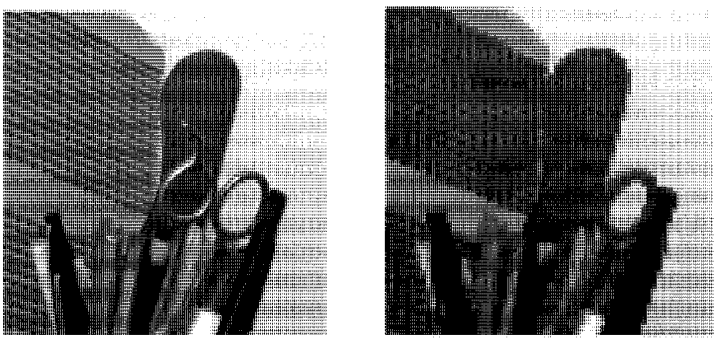


Figure 1-14. Loss of spatial detail from subsampling.

1.3 Color Perception

Color perception in humans is often misunderstood. Because of this the transition from monochromatic imaging to color imaging is sometimes misrepresented. Our visual system processes three bands of the electromagnetic spectrum independently and so the system incorporates a type of multisensor fusion. The region that these (contiguous) bands cover constitutes the visible spectrum for humans, which extends roughly from 700 nanometers of wavelength to 400. At the short end of wavelength we see shades of blue; at the long end we see red. Going further on the short end leads us into ultraviolet and on the long end to infrared. The photopic response curves for humans is shown in Figure 1-15. Researchers believe that this response is best to maximum contrast of yellow and red objects against the green hues of the tropical forest canopy.

We see colors as additive color mixtures of the primary colors red, green, and blue. The color bands mix to yield variations in color perception across the set of red, orange, yellow, green, blue, indigo, and violet—the colors of the rainbow. The limits of our perception in this regard are clear from the so-called tri-stimulus diagram, shown in Figure 1-16. The diagram represents all additive mixtures of colors using equal intensities of red, green, and blue light. At each vertex is a pure color, and points within the triangle represent mixtures of the three. Points on the sides of the triangle are combinations of two colors. The area under the curve shows what we are capable of perceiving based on our response curves (Fig. 1-15), while the crosshatched area is the region of

physically realizable colors that we cannot see. The bisecting lines meeting at the center show the point of white light perception. You may have been in a theater and observed that the white light on stage has been generated by a set of red, green, and blue floodlights.

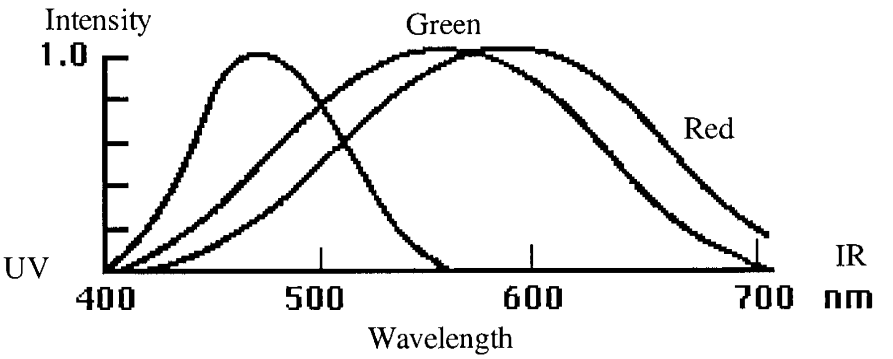


Figure 1-15. Human color response.

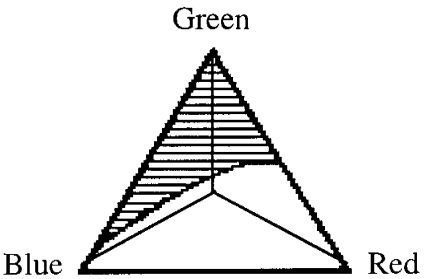


Figure 1-16. Human color perception tri-stimulus diagram.

For machine vision, color processing is only useful as a means of acquiring greater information beyond monochromatic imagery. Few algorithms extend directly from grayscale to color, and the restrictions on color processing because of specialized equipment and substantially increased computational demands add to the overall concern about its usefulness. From a multisensor fusion standpoint, however, color processing is invaluable in many recognition tasks, particularly when color yields a strong feature invariant. Because of the information richness that color imaging can provide, we, and ultimately the machine, can distinguish more objects or regions in a color image than a monochromatic one.

1.4 Motion Perception

The perception of moving objects, tracking of moving objects, and the ability to see still scenes from moving image sequences are all related to motion perception. The idea of flicker fusion and critical fusion frequency were discussed earlier. If you have movie or animation software, you can determine your rate of fusion by varying the frame rate, or frames-per-second (fps) speed, of a movie that you produce. When you stop seeing individual frames, you have reached your fusion rate. The fusion rate is a function of time delays and integrations that occur within the HVS. Machines have no fusion rate; the frames in an image sequence are acquired at a rate determined by the hardware. One can purchase hardware that will acquire images fast enough to stop the flight of bullets. Processing the images at the same rate, however, is another matter entirely. High-speed image sequences are processed off-line, one frame at a time. Some operations can be processed in *real time*, but they are limited.

The detectors in the retina are chemical-based and have low persistence. If this were not the case, we would maintain the same image longer and so miss a large amount of motion that takes place in the natural world. The sensing system acts something like a CCD in that the image must be continuously refreshed and sampled. As previously mentioned, the eye is in constant motion due to involuntary movements called saccades, and these motions yield the necessary sampling rates. In short, the HVS is a dynamic variable-rate sampling system. This form of sampling implies that the system changes its rate of data acquisition based on the information content of the signal. The most compelling reason to use this approach is to gain greater processing capability given limited processing resources. Therefore, the HVS uses autonomic musculature and structure to reduce processing load on the brain when information is scant and increases data flow where information content is rich. To understand this more clearly, recall that the sensor density is greatest at the fovea, at the optical center of the eye. The HVS uses a combination of mechanical positioning, variable sensor density and data reduction at the sensor (the ganglia of the retina) to form a complex coded image that is transmitted to the visual cortex of the brain. Thus, the complexity of this system illustrates the difficulty of producing a machine vision system similar in capabilities to that of the human. This is not to say that we should seek a full biomimetic duplication of the HVS because compromises were made as a result of developmental optimization and sequencing. Hence, the system is not perfect and in many cases will not be the optimal model for a machine system. Nevertheless, we may learn a great deal of what is possible in a machine vision system from study and observation of the biological.

REFERENCES

1. S. Zeki, *A Vision of the Brain*, Blackwell Scientific Publications, Oxford (1993).

2. R. Boynton, *Human Color Vision*, Holt, Rhinehart and Winston, New York (1979).

CHAPTER 2
IMAGE PROCESSING

Image processing is the manipulation of images using computer algorithms to enhance, restore, and understand the information contained in them. Images can be treated as two-dimensional signals, so a large number of signal-processing algorithms may be employed. To study image processing, we start by defining the characterization of images as two- dimensional signals and conclude with how these signals can be parameterized in preparation for machine vision analysis.

2.1 IMAGE CHARACTERIZATION

An image, in the context of image processing and machine vision, is a two-dimensional array of picture elements, or *pixels*. These elements are typically derived from the sensing of scene data using a camera or scanner. The value of a pixel is determined by the system used to sense the object. For example, in a visible spectrum, monochromatic camera, the pixel represents light intensity data reflected from objects within the scene. A laser radar supplies pixels that represent the reflected energy of the laser employed in the radar, pixels that represent range to objects, or, in the case of the Doppler shift of the returning radar beam, pixels that represent the motion of objects. The pixels of an image can be indexed as values of a two-dimensional function, $f(x,y)$, as shown in Figure 2-1.

Typically, the upper left-hand corner of the image is taken as the origin, with x identifying the row and y the column indices for a pixel. Each pixel, $f(x,y)$, represents an irradiance as reflected from the various objects and regions seen in the picture. The total number of pixels in the picture is determined by the sampling of the image while the range of values that a pixel can take is determined from the quantization. Both of these terms, sampling and quantization, are discussed in the next section. Any process such that $g(x,y) = O[f(x,y)]$, where $O[\bullet]$ is some operation, will produce a new image. Determining what $O[\bullet]$ will produce the required result is the basis of image processing.

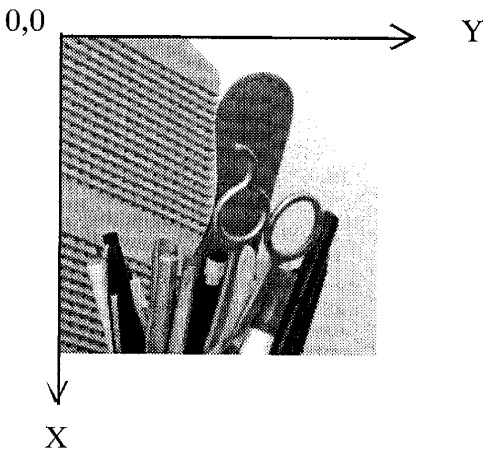


Figure 2-1. Image function and geometry.

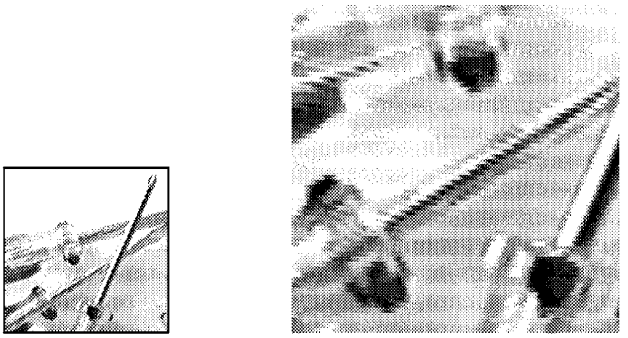
2.2 SAMPLING AND QUANTIZATION

The number of pixels in a given area defines the resolution of the image. The more pixels per unit area, the higher the resolution. However, the actual size of a pixel is determined by the medium used to display it. For example, consider pixels that represent a single value of one or zero. A black and white printer can print pixels of this sort easily by printing a black dot when the pixel is zero or leave a blank when it is one. The higher resolution printer will print finer lines and consequently be capable of greater detail. Alternately, the same image can be printed with different resolutions with the same printer. The sentence below on the left is twice the resolution in dpi than that on the right.

Sampling loosely refers to the number of pixels in a given image. The terminology may be somewhat misleading because in signal processing sampling means "samples per unit time." In the same sense, the image is "samples per unit distance." However, as shown above, the *resolution* of the display is what will determine the *resolution* of the image that is output—in this case the resolution of the printer used to produce this page. Images are described in terms of row-column dimension and these dimensions generally follow binary conventions such as 256 × 256, 1024 × 1024, etc. Images that are 512 × 512 are popular because they are close to the resolution of NTSC television in the United States.

Figure 2-2a shows a 256 × 256 pixel monochrome image printed at 300 dpi. Figure 2-2b shows a subpicture that is 64 × 64 pixels printed at the same resolution, 300 dpi; however, the actual resolution of the larger subimage is 150 dpi. In other words, the smallest black dot that can be resolved from the first

picture is 1/300th of an inch, whereas the smallest dot resolvable from the second picture is 1/150th of an inch.



(a) (b)

Figure 2-2. (a) 256 × 256 image, 300 dpi, (b) 64 × 64 image, 150 dpi.

The range of values for a pixel is determined by its *quantization.* Quantization of monochrome images is described as *grayscale*, or the scale of values that represents a pixel range of grays that take it from white to black. A grayscale with a span of 0 (black) to 255 (white) is given in Figure 2-3 below. A pixel takes on a graylevel, from the grayscale, which is the intensity value of the pixel.

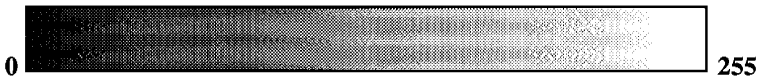


Figure 2-3. Grayscale.

For a grayscale of 0 to 255, one 8-bit byte of computer memory is required to store the pixel value. For a grayscale image of 512 × 512 pixels, 262,144 bytes, or 256K are needed. If we have only one bit per pixel to work with (black or white), we can synthesize grayscale using either a *dithering* process or *half-toning*. An image that is represented by single-bit pixels is called a *binary* image. All of the images in this text have been produced by half-toning, which is where the "darkness" of a pixel is determined either by size or by clustering. Dithering is a technique by which a grayscale pixel is represented as a cluster of binary pixels in patterns of varying density. The eye integrates these patterns to yield variations of gray. Dithered and halftone images are not used for machine vision analysis because information is lost in the process and *pseudocontours* may be introduced into the image. A pseudocontour is an edge that appears

because there is not enough sampling to accommodate a smooth transition of graylevels.

Images can have greater quantization and hence a wider range of grayscale. This depends on the digitizer used to evaluate the input signal forming the picture. The 8-bit (single byte) pixel is used because of our human limitations; we cannot distinguish between two adjacent pixels that differ by one graylevel. In some applications, a 12-bit per pixel quantization is warranted. The computer can certainly analyze this entire range, so caution must be exercised when developing a system to prevent human limitations from distorting the design.

Color images can be represented in various ways and are usually constrained by the display or print system being used. Of interest to machine vision are *true-color* images, also called 24-bit color or *RGB* images. A pixel in a true-color image is actually represented by three pixels: one red, one green, and one blue (see Figure 2-4). Each of these pixels has a range of intensity like the grayscale, but instead we might say "redscale" or "bluescale." When the three pixels appear on a display, the additive color mixture (see Section 1.3) allows the eye to perceive a unique color. When printed, the mixing of primary color inks to yield color is called *subtractive color mixture*. With additive color mixture the primaries were red-green-blue (RGB) while the primaries for subtractive color mixture are cyan-yellow-magenta (CYM). Machine vision treats color images of this sort as multispectral representations. A unique color to a human is simply a vector of values to the computer. The computer can identify objects with specific colors rapidly, as we shall see in Chapter 5.

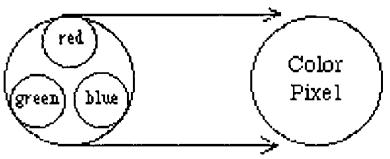


Figure 2-4. Color pixel formation.

True color image files can be huge. For example, a 640 × 480 picture consumes 900K of storage. Various techniques of compressing image files are available, but for machine vision purposes, lossless schemes are recommended. For example, *JPEG* compression uses a lossy method that takes advantage of the human inability to discern small color variations in high-resolution pictures. As a consequence, JPEG compression discards this data and so a compression is achieved. The discarded data would be perceived by a machine vision system (the JPEG algorithm detected it!) and may have importance to the program goal.

2.3 SPATIAL FREQUENCY PROCESSES

In the previous section we discussed sampling and resolution and stated that an accurate description of these would include the concept of "samples per unit distance." When we discuss single-dimensional signals, we typically speak in terms of frequency that is measured in cycles-per-second, or Hz. Similarly, we can describe images in terms of *spatial frequency*. To see this, consider Figure 2-5. The sinusoid on the left is half the frequency but the same magnitude of the sinusoid on the right.

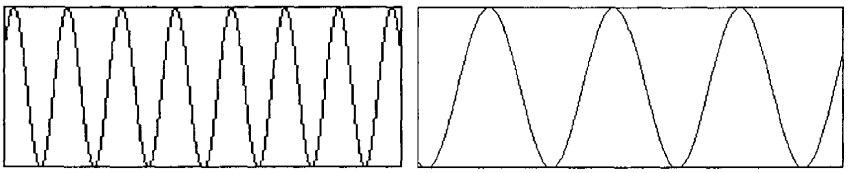


Figure 2-5. Single-dimension sinusoids.

Now consider a sinusoid in three dimensions by letting the magnitude of the sinusoid be represented by brightness, such as illustrated in Figure 2-6. The graphic of a three-dimensional sinusoid in the left column of the figure is mapped to the two-dimensional image on the right, so that in this form the peaks of the sinusoid will appear as light lines and the troughs as dark. Now we can discuss images in terms of spatial frequency. Recall that the thinner the line in an image, the higher the resolution and the spatial frequency. Since lines in an image can run in all directions, it is sometimes difficult to visualize the spatial frequency content of a picture. The more detailed an image is, the higher the spatial frequency content it will have and the more resolution a system (acquisition, storage, or display) will need to resolve the image completely.

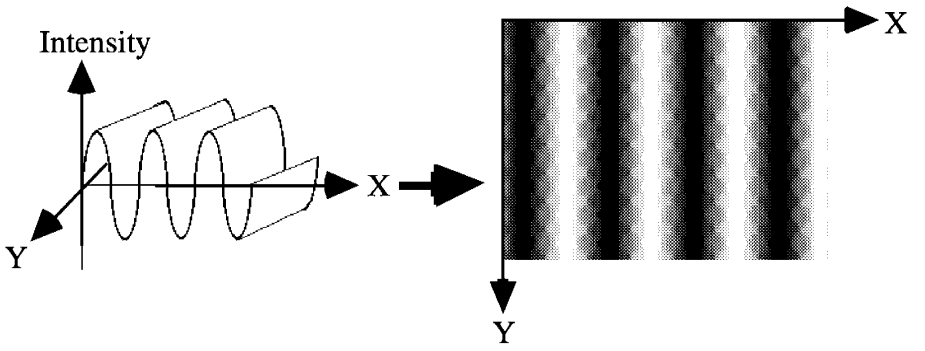


Figure 2-6. Three-dimensional sinusoid as image.

For a one-dimensional signal in-time, we can decompose the signal into signals of fundamental frequencies. The set of fundamental frequencies must have the mathematical characteristic of being a *basis* set. One such basis set is comprised of sines and cosines and the method by which a set of coefficients is resolved is called the Fourier transform. The two-dimensional (2-D) Fourier transform can be applied to images to resolve the spatial frequency spectrum. Conversely, the spectrum can be reconstituted back to the original image using the inverse Fourier transform. The usefulness of the 2-D Fourier transform to machine vision work is somewhat limited to filtering operations to improve image quality prior to higher-level processing, although some techniques exist that use the transform for specialized feature extraction. For more complete discussions of the transform, a number of good treatments can be found in the literature.[1,2,3]

2.4 NEIGHBORHOOD PROCESSES

As discussed, the Fourier transform is very useful in the filtering of images, or when one wishes to extract or detect features that are characterized by spatial frequency. For example, recall that thin lines in an image are represented by a high-spatial frequency. If we can remove the low spatial frequencies from an image, the high frequency lines should be enhanced. The watch image on the left in Figure 2-7 was Fourier transformed. The components corresponding to low spatial frequencies were removed and the filtered result was thresholded; pixel values below a particular level were discarded. The edge image to the right was obtained. Now consider the watch image of Figure 2-8, where the edges have been enhanced by the use of a spatial filter mask by employing a discrete convolution operation.

Convolution means "to wind or coil together" and the algorithm basically does that between two images. The convolving occurs as a sum-of-products operation between the pixels of one image and that of another. Application of a discrete convolution algorithm yields a spatial filtering operation. The algorithm performs a multiplication in the spatial frequency domain between two functions without requiring a Fourier transform. The Fourier transform is computationally complex and places great demand on computing resources. The discrete convolution can achieve similar effects to the transform at a substantially reduced computational cost. One of the properties of the Fourier transform is that multiplication of two transformed functions yields a convolution when the result is inverse transformed. This convolution takes place in the spatial domain (not the spatial frequency domain), hence a discrete convolution results in a spatial filter. Spatial filters use a mask, or small image, that is convolved across the target image to achieve the desired effects. The values of the mask elements determine what effect the mask will have on the image.

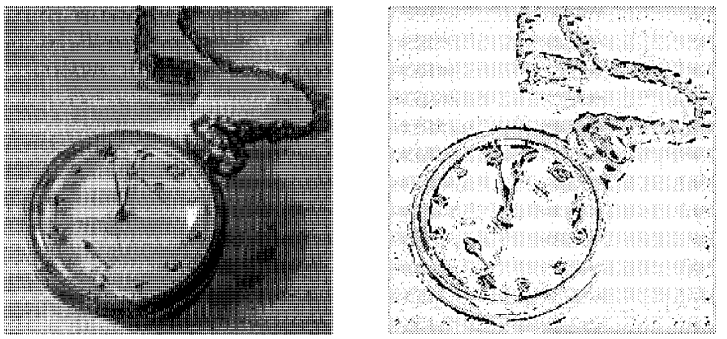


Figure 2-7. Edge detection/enhancement using high-pass filtering.

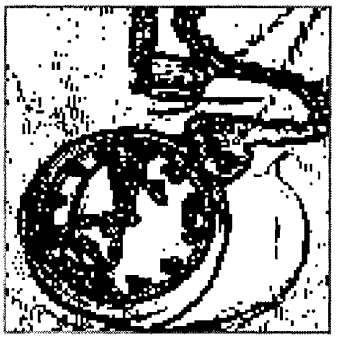


Figure 2-8. Edge detection/enhancement using spatial filter.

In Figure 2-9 an image is shown with a mask superimposed on it and the detail of a 3 × 3 mask with the mask values (pixels) identified by the variables *a-i*. Assume that when the mask is aligned over an area of the image that the image pixels are identified by the variables a_i to i_i. To compute the discrete convolution, we use the following expression,

$$e_c = a \cdot a_i + b \cdot b_i + c \cdot c_i + d \cdot d_i + e \cdot e_i + f \cdot f_i + g \cdot g_i + h \cdot h_i + i \cdot i_i$$

where e_c is the pixel in the resulting convolution at the same index as the image pixel under the center of the mask. Note that the equation computes the sum-of-products between the mask and the image pixels below it. The summing and shifting can be thought of as a *copying* operation where the mask is copied at each pixel of the image and weighted by the values of the image pixels in the neighborhood where it is applied. As seen in Figure 2-8, the extraction of the edges of the objects in the image is of great value in machine vision work when one is trying to determine which objects are being seen by a system and where those objects are in the field of view.

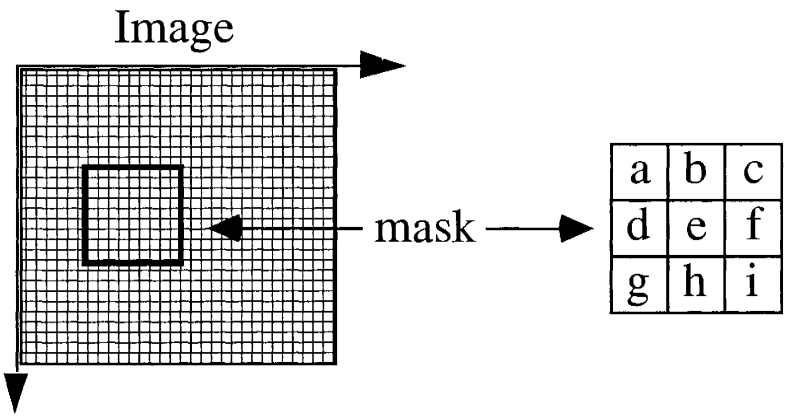


Figure 2-9. Convolution mask.

The discrete convolution is a neighborhood process, in contrast with a point process. The distinction is shown in Figure 2-10, below, where (a) shows that a point process effects an operation, P[•], that maps pixel to pixel, while (b) shows a neighborhood process where an operation, O[•], combines a clustering of pixels to produce a result.

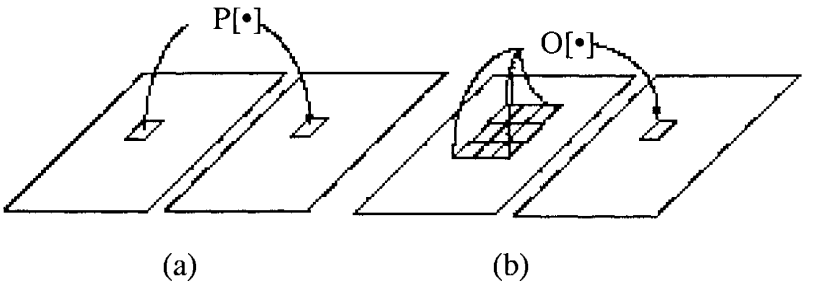


Figure 2-10. (a) Point and (b) Neighborhood processes.

There are basically three types of spatial filters:

a) Low pass,
b) High pass, and
c) Nonlinear.

Low- and-high pass filters are easily characterized. They are applied using the discrete convolution described above. A low-pass filter mask will have only positive values and a high pass will have negative values. This is easily explained intuitively by the fact that the discrete convolution operation is a spatial sum-of-products (see equation above for e_c). Recall that the operation copies the mask across the image, weighted by the image pixels. If the mask is

positive, it will increase the overall pixel values and tend to spread the effect of each pixel, thus widening lines and decreasing the overall spatial frequency—which is the definition of a low-pass filter. To see this, consider Figure 2-11 below. The figure shows three images, the first is a conch shell rendered in grayscale, the second the shell after a low-pass spatial filter has been applied, and the third is a three-dimensional rendering of what the filter looks like—something of a peak. This filter uses a 7×7 mask with the following values:

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

From the figure you can readily see that the effect is one of smoothing. The filtered conch is very fuzzy and indistinct.

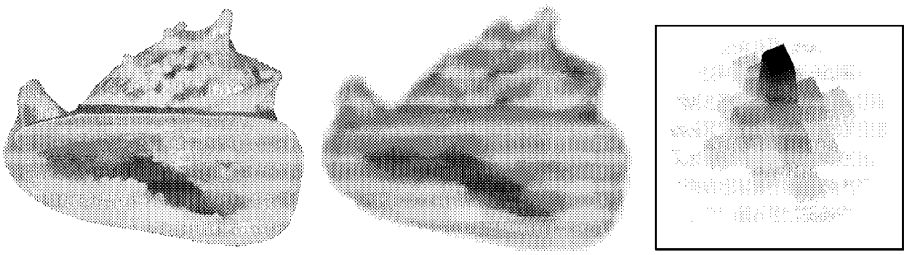


Figure 2-11. Conch shell, low-pass filtered conch, low-pass filter.

Now consider the same conch filtered with a high-pass mask, as shown in Figure 2-12. Now the edges, the strongest edges, have been isolated and enhanced. These edges constitute the outer contour of the shell. The filter used is a 5×5 mask, but now the mask has negative values, as shown below:

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

This mask can remove values, thus accenting regions of rapid change in the image, such as at the boundary of edges. At this point it should be clear that a

mask can be of any size and the mask need not be symmetric or odd. Symmetric masks act in all directions with respect to edges in the image and odd-dimensioned masks make for simpler programming. The following masks are nonsymmetric and will enhance diagonal edges:

```
-2 -1  0     0 -1 -2
-1  0 +1    +1  0 -1
 0 +1 +2    +2 +1  0
```

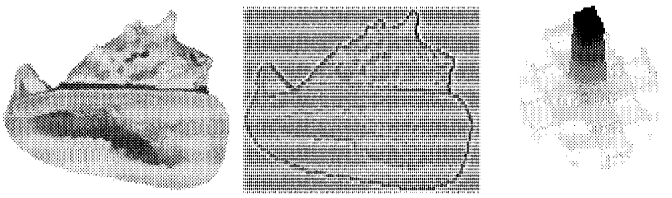


Figure 2-12. Conch, high-pass filtered conch, high-pass filter.

Nonlinear spatial filters do not utilize the discrete convolution algorithm. Instead, they generally involve a statistical operation (such as computation of the median) or are decision directed. A decision-directed filter is one that changes filtering behavior based on what it senses in the content of the image as it is processed. Nonlinear filters are most useful in the enhancement of images, so it might be assumed that they are not of much use in machine vision. This is somewhat true; however, many images require preprocessing to remove noise that might otherwise disturb the analysis. The dilemma in filtering noise is, of course, not to filter information necessary to the recognition. This topic will be revisited later when we discuss objects and regions in Chapter 5.

2.5 POINT PROCESSES

Recall Figure 2-10(b), the diagram illustrating the nature of a point process. Here, pixel values are operated on by P(•) and are mapped to single corresponding pixels in the resultant image. The most common of the point processes is histogram equalization. The *histogram* of an image is the probability distribution of its pixel values. To generate a histogram, we count the number of pixels at grayscale zero, then the number at grayscale one, etc., until we have counted the number of occurrences throughout the entire range of values. We then normalize the counts by dividing each by the total number of pixels in the image. Figure 2-13 shows a mountain with snow and shadowy trees in the foreground. Also given is the histogram. Note that the majority of pixels have a value of less than 128, indicating a "dark" image, as we see.

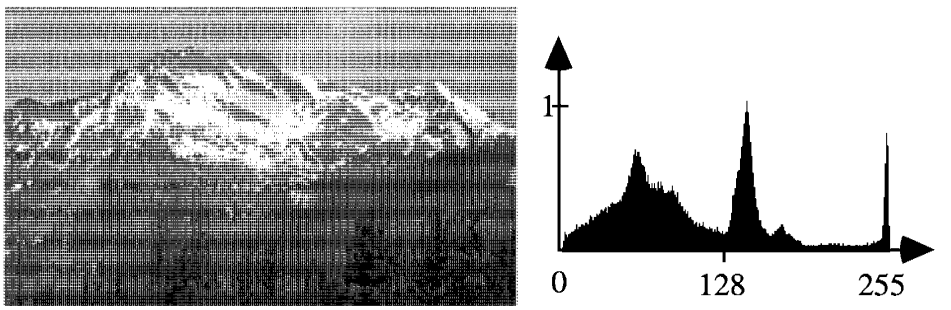


Figure 2-13. Mountain image and histogram.

Histogram equalization is a process by which we treat the histogram as a probability distribution and convert it to a uniform distribution by integrating. This is a well-known result from statistics, but for our purposes we simply accumulate histogram values and use the accumulations to remap the pixel intensities. This process has the effect of "flattening" the histogram, thus increasing the contrast of the image. Figure 2-14 illustrates this process with the mountain image. Note that the sky is considerably lighter and the trees are somewhat more distinct.

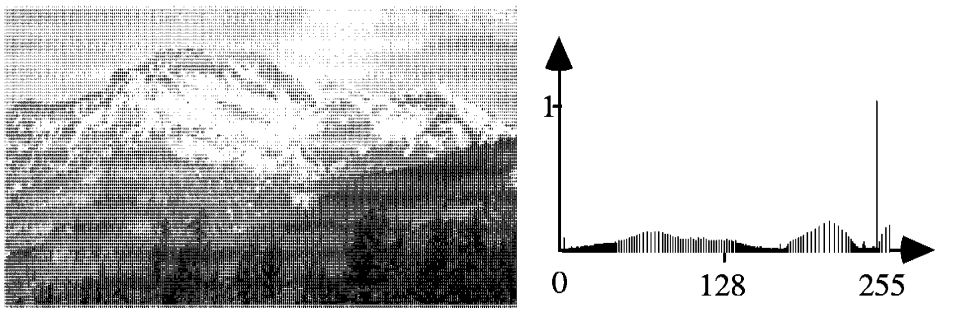
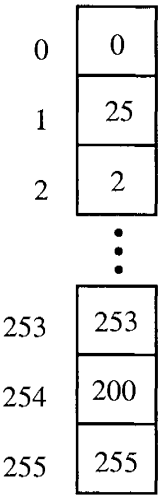


Figure 2-14. Equalized mountain image and histogram.

The equalized histogram *is not completely flat* due to accumulated errors in the summing of the discrete levels (integration). Nevertheless, a contrast enhancement was achieved. When operating with hardware, the equalization can be applied to a *look-up table*, or LUT. A LUT maps a pixel value to a display value. For example, if our equalization (or other histogram operation) called for pixels of value 254 to map to pixels of value 200, the value of the LUT at location 254 would be displayed as 200. Think of the LUT as a single-dimensional array with elements corresponding to the number of graylevels possible, as depicted in the graphic to the left. Here we see that the LUT maps input pixels of value 0, 2, 253, and 255 to the same value on output, but pixels of value 1 are mapped to value 25 and pixels of value 254 are mapped to value 200. What this implies is that by changing the mapping values, we can redistribute the grayscale values of pixels in an image in any way that we choose. Mappings of this sort can be inverted—we can simply map the pixels back through the LUT to return to the original distribution for point operations. This is not the case with the neighborhood operations in which the operation cannot be reversed. From the standpoint of machine vision, a point process can be a useful preprocess step before more complex analysis routines. We will see in Chapter 5 how histograms play an important role in the thresholding of objects and regions.



2.6 IMAGE PROCESSING AND MACHINE VISION

To try and cover all areas of image processing in one short chapter of a machine vision text would be foolhardy. The annotated bibliography contains a number of excellent references to expand on what has been discussed here. One need not be an expert in all aspects of image processing to be successful in addressing machine vision problems. Machine vision is used to understand the content of images while image processing improves them. This is an important distinction, because for almost all applications, unless one has control of image capture, one can ignore compression, restoration, and reconstruction algorithms.

Image processing can be partitioned into three levels, as listed in the chart below:

Low Level	Mid-Level	High Level
histogramming filtering warping and registration reconstruction compression/decompression	edge detection thresholding segmentation	representation recognition interpretation

The distinguishing factor in this partitioning is that processing at each level results in a unique outcome if we restrict the definition of an image to that of a natural scene. The output of low level operations is an image— hopefully an improved image—but an image nevertheless. At this level, histogramming can take a poor contrast image and improve the effects of bad lighting; warping and registration algorithms can compensate for poor camera placement or the curvature of the earth in images taken from satellites; and filtering can remove noise. Mid-level operations produce a two-dimensional data structure. For example, the result of edge detection may look like an image, but it is no longer a natural scene. Finally, high- level operations discard the image completely and generate a data structure that describes the content of the scene—this is the basis of machine vision.

In summary, the following two points can be made: (1) image processing is useful for the preprocessing of images to enhance the later stages of processing for machine vision purposes, and (2) one must use caution when preprocessing so that data are not lost. An example of these issues is shown in Figure 2-15. Since machine vision problems typically involve locating an object and making a decision of some sort about it, the image processing operations shown in the figure could either enhance or detract from a recognition operation. The edge detection of the sharpened image (bottom, middle) gives the least fragmented result, particularly in the case of the bolt at the bottom of the image and the nut to the left of it. It should also be clear from this processing that resolving the difference between the three bolts is nontrivial. If we measure the length of each bolt in terms of pixels, we will discover that each differs only by two pixels. Lighting can cause a difference of this many pixels in a recognition scheme. Since the two nuts are oriented the same way, it is very simple to find them by using a correlation mask. However, if they were not oriented alike then their recognition would require more processing in an attempt to detect and identify all possible orientations.

The final point that must be made is that image processing techniques can be used in a great number of ways to extract information from a given scene so that the machine vision algorithm can discriminate between objects the scene contains. Nevertheless, the outcome of processing is highly dependent on lighting conditions, resolution, and complexity of the objects that must be recognized. These issues and others will be explored in Chapter 4 along with approaches that can be used for machine vision processing.

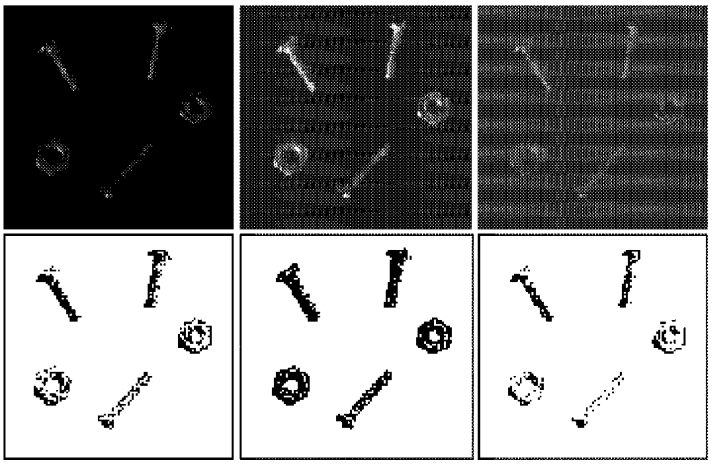


Figure 2-15. Upper row: (left) original image, (middle) image after sharpening filter, (right) image after smoothing filter. Bottom row: edge detection results for each of the upper-row images. Note improvement after sharpening and loss of detail after smoothing.

REFERENCES

1. R. Tolimieri, M. An, C. Lu, and C. S. Burrus, *Algorithms for Discrete Fourier Transform and Convolution*, 2nd edition, Springer Verlag (1997).

2. W. L. Briggs and V. E. Henson, *The DFT : An Owner's Manual for the Discrete Fourier Transform*, Society for Industrial & Applied Mathematics (1995).

3. K. Morita, ed., *Applied Fourier Transform*, IOS Press (1995).

CHAPTER 3
COMPUTER GRAPHICS

The field of computer graphics involves the creation of images and image sequences from data descriptions and is closely allied with fine arts and media studies. The importance of computer graphics to machine vision is that the reduction of a natural or real-world scene to a graphic representation can be a powerful step toward the understanding of the scene content by the computer.

3.1 DEFINITION

Where image processing seeks to evaluate and process images of scenes and objects derived from data sensed in the real environment, computer graphics attempts to create scenes and environments from synthetic data structures. Consider a sequence of image processing activities, where a natural scene is processed, such as in Figure 3-1(a). Here the image is that of an aircraft landing. We see the sun and a portion of the top of a security fence. In 3.1(b), the image has been edge detected (see Section 2.4) and is no longer an image of a "natural scene," but has become a two-dimensional data structure that identifies the spatial position of important edges in the image.

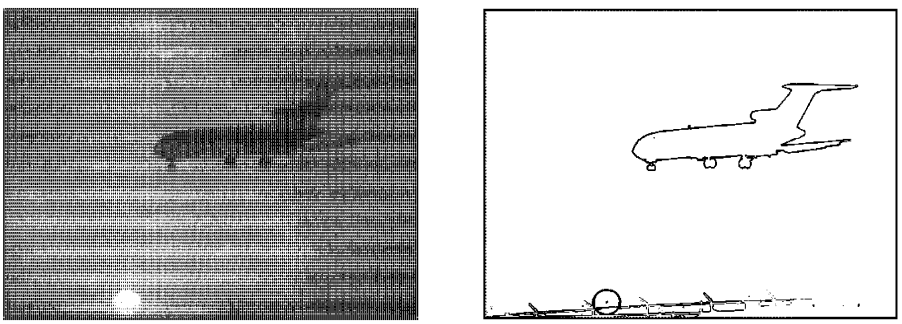


Figure 3-1. a) 727 jet landing, b) edge-detected jet.

With further work, we can process the image to the point where all that is left is a data structure, such as the one listed in Figure 3-2. Here the content of the image has been reduced so that we no longer have an image at all.

However, when given the description, we can, with some imagination, visualize the original scene. Of course, if all one has is the description below, one might construct an image of a 727 with a horizontal view and at the coordinates specified, but the image created would not be the same as the one we started with. We have neglected to describe the exact orientation of the jet, the shading and lighting conditions, the fence description, etc. One of the major goals of machine vision study is to be able to reduce an image to a description such as that given in Figure 3-2. In contrast, computer graphics seeks to take a description and derive a realistic scene.

Image: 640 x 480 pixels, 256 graylevels

Object A:	Jet, 727	ObjectB:	Sun
View:	horizontal	View:	direct
Coord:	(365, 276)	Coord:	(188, 23)
Area:	16,815 pixels	Area:	1,334 pixels
Mean:	203.77	Mean:	255

Figure 3-2. 727 Image description.

This is the contrast between image processing and computer graphics. With image processing, one starts with a natural scene derived from a sensor system and produces a transformation. In the case of machine vision, the ultimate goal is to produce a data structure that describes the natural scene so that some sort of intelligence can be derived from the description. In contrast, computer graphics starts with a description and attempts to derive a realistic-looking natural scene. It is because of this duality of purpose, albeit in different directions, that an understanding of computer graphics terminology and methods is important to the student of machine vision.

3.2 GRAPHIC OBJECTS AND PROCEDURES

Graphic objects are formed by drawing, or by recreating, a pixel value over a path or contour. The simplest graphic object is a point, and this, of course, leads to the next object, a line, and so on. In the early days of computer graphics, just plotting data on a raster screen was difficult. Today most word processors contain graphics editors so that a drawing can be created and placed during document creation. To produce overlay graphics on a computer image, the pixels of the image that will represent the graphic objects must be modified. Consider the image of the jet in Figure 3-1(a). Let's say that we want to show a line that goes from the center of the jet towards the edge of the image at the angle of approach, as shown in Figure 3-3. The line is drawn by changing pixels starting at the coordinates of the center of the aircraft and leading in a straight line to the edge of the image to white. This is easily accomplished with a line drawing routine based on the two-point form for the equation of a line.

There is a slight dilemma in drawing lines that is readily seen in the figure. We are dealing with a discrete coordinate system and lines may not appear smooth if the sampling is coarse. The jaggedness of the line is called aliasing, in the same way that the term is used when discussing sampling. This effect can be minimized by algorithms that compute the best fit of the given pixel sampling to the ideal line between two points. These algorithms are called *antialiasing* or *dejagging* techniques.

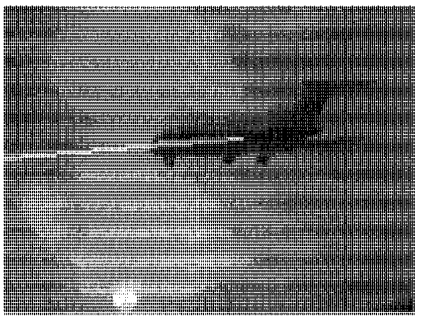


Figure 3-3. Graphic line overlay.

The fundamental graphic objects are points, lines, ellipses, rectangles, polygons, and free-form contours. Ellipses and squares are special cases of circles and rectangles. Just as we established images as represented on a Cartesian plane, so also can computer graphics be represented. Any geometric object that can be mathematically represented on that plane can be recreated by the computer using pixels as points.

When perspective is incorporated into line drawings, three-dimensional representations can be created. Consider a circle that is extruded along a line passing through its center creating a cylinder, as shown in Figure 3-4. When the cylinder is viewed at an angle, the circular ends become ellipses. As one end of the cylinder recedes from the viewer, that end collapses to a point. It is beyond the scope of this book to explore 3-D methods in graphics; however, 3-D plays an important role in machine vision analysis as revealed in Section 3.3.

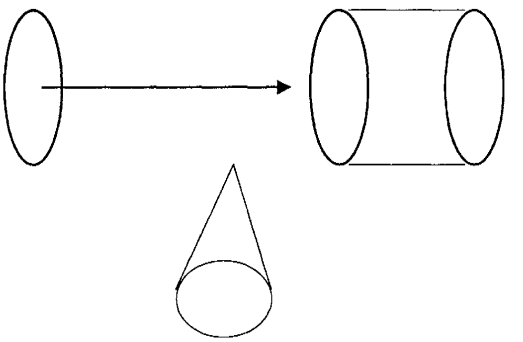
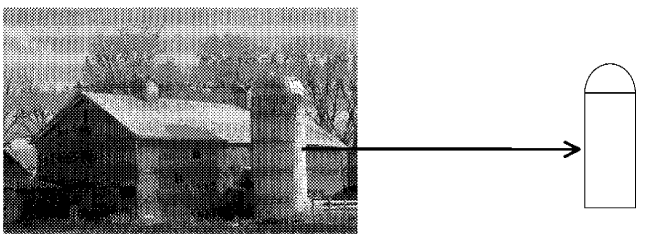


Figure 3-4. Extruded circle and receding cylinder.

Image processing hardware often includes a *graphic overlay* channel. This is an extra frame buffer that overrides the pixel values of the main buffer when the image is displayed. Thus, graphic objects can be used over the image being worked without disturbing the pixels of the image. This permits the graphics to be changed or animated without having to restore the original pixels.

3.3 USEFULNESS TO MACHINE VISION

The relationship of computer graphics to the field of machine vision is found in the correspondence between the geometric properties of computer graphics and the geometry of objects in scenes. If an object in a real scene can be reduced to a simple graphic object, or collection of graphic objects, then the identification of the original object can, in most cases, be greatly simplified. Consider the barn and silo image of Figure 3-5(a). The silo is essentially a cylinder with a hemisphere on top. In 2-D, it can be represented as shown in Figure 3-5(b), a rectangle and a half-circle.



(a) (b)

Figure 3-5. Barn and silo.

For now, we can dispense with the difficulty of extracting the graphic of the silo from the farm scene as that is covered later. Instead, we can concentrate on the problems of scale and rotation. It is relatively simple to look for a particular shape at a specific size and orientation, but the search becomes difficult if the objects can have large variances in these parameters. When the problem is one of factory automation, where the area of operation is fixed, then these problems are of less consequence. In factory automation the ability to control the lighting, camera position, orientation, and the positioning of the objects under scrutiny, is a distinct advantage.

Figure 3-6 shows the setup for a factory inspection system of parts on an assembly line. One can control the following:

- Camera: position, lens, filters, and resolution.
- Lighting: eliminate shadows and enhance contrast to camera.
- Conveyor: background and travel speed of parts under camera.
- Parts: orientation.

This short list of controlled parameters can greatly simplify the machine vision task by constraining the *geometry* of the problem. In short, controlling the environment of the machine vision problem reduces it to a graphics problem. Increasing control of the image acquisition parameters, if you will, from the graphic representation to the actual scene perceived by the camera decreases the perceptual "distance." Standard and well-studied algorithms can then be employed to evaluate the acquired image.

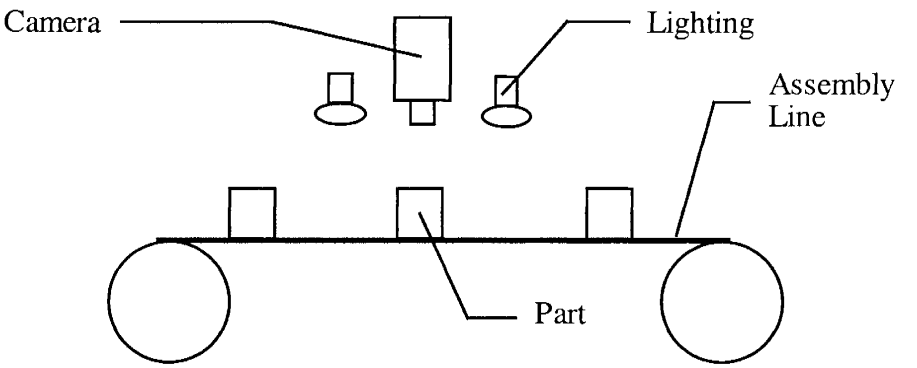


Figure 3-6. Assembly line set-up.

In the scenario of Figure 3-6, the only parameters not mentioned are speed and computing power. The speed of task accomplishment will depend directly on the resolution needed and the amount of detail in subtasks that the machine vision algorithm suite must evaluate. These issues are covered in greater detail in later chapters. To summarize the importance of computer graphics processing to machine vision: the reduction of real scenes (whether from factory assembly lines or military surveillance cameras) to graphic representations is a powerful methodology for machine comprehension of images.

The intersection between computer graphics and machine vision narrows as time passes. Large-scale vision systems employed in factory automation scenarios have major components such as graphical user interfaces (GUIs) that are developed from computer graphics methods. These systems also use geometric constraints such as view and perspective in the analysis and evaluation of moving objects, making the understanding of computer graphics processes a necessary element of machine vision study. Finally, machine vision systems are increasingly being paired with robots. This requires a high degree of coordination between the internal geometric representations of the workspace in the vision system and the actual geometry of the workspace that the robot is embedded in. The vision system is operating with a virtual space that must correlate closely to the actual space if the decisions made by the vision system are to be effective in controlling the robot.

CHAPTER 4
MACHINE VISION

Machine vision is the computer analysis of images with the intent to discover what information the images contain. Other terms such as image understanding and computer and robot vision are synonymous with machine vision. If any differentiation exists, it is related only to a source research area. For example, the term *image understanding* is typically used by image and signal processing researchers. *Computer vision* hails from computer science and artificial intelligence studies, and the term *robot vision* comes, naturally, from robotics.

4.1 GOALS

Ideally, we would like to have a machine vision system that displays the same capabilities as the human visual system. These capabilities can be summarized as follows:

- perceive lightness and color of surfaces under a variety of illuminations.
- detect significant changes in intensity and perform 2-D segmentation into useful regions.
- infer 3-D structure of scene surfaces from a variety of monocular cues and from a sequence of stereo or motion images.
- organize the surfaces and regions into objects of interest.
- generate descriptions of objects and recognize them among a potentially large class of objects.
- make nonvisual intelligent inferences about the scene based on the visual processing (abstraction).

In previous chapters, we discussed the problem of uneven illumination and how the human visual system adapts to a wide range of lighting conditions. We also mentioned briefly the complexity of 3-D representations and have loosely discussed the concepts of object and region extraction. Nonvisual inference is more closely aligned with *machine intelligence* studies. However, since the sensory system of the human is dominated by the visual subsystem,[1] the importance of drawing inference from visual cues is equally at home in machine vision. Overall, in machine vision we strive to generate a complete understanding of the target image and what it contains, and so the methods of machine reasoning play an important role in this study.

All that is required for the study and application of machine vision is a computer system. Ideally, however, we want to have image capture and display facilities. These capabilities increase the cost and complexity of the system but are requirements of modern-day machine vision analysis. If image sequences are needed, then the cost and complexity rises significantly. The issues surrounding image sequence study are discussed in Chapter 7.

Before selecting specific machine vision goals, it is necessary to establish a domain of operation for the analysis. This domain can be as restricted as the factory assembly line scenario shown in Figure 3-6, or as unrestricted as a camera mounted to a completely autonomous robot. In the case of an autonomous vehicle, one can retract the visual domain to paved roads with the assumption that the vehicle will not operate on other surfaces. Even this simplification adds to the success probability of the overall system. Domain constraint may also include the image sensor parameters such as sampling, quantization, field of view, and wavelength (or band). In complex systems, multiple wavelengths are used and the result is a system that incorporates *multisensor fusion.* The constraint in these systems is the synergy and contrast that often exist between different sensor wavelengths as applied to specific problems.

The goals of a machine vision system will become, whether the user wants them to or not, tightly coupled to the technology of both the target application and the requirements of analysis. For example, the speed of analysis of the system, which becomes very important in many factory scenarios and critical in a high-performance aircraft, will limit resolution of the system, increase complexity of the subunits, and require large computing resources. The complexity of the objects that the system must evaluate will also drive the resolution demands. A system can detect and evaluate wrenches on a conveyor belt much easier than it can detect anomalies on a printed circuit card. Also, if the target objects and/or the system itself are moving, the vision system will be constrained by the amount of time needed to acquire a useable image and by the time that it has to make a decision or issue a report to another system. These issues are somewhat empirical, however, good design planning can prevent complications that arise because of hardware limitations. Appendixes A and B review software and hardware needs for machine vision and offer hints on selection of packages and equipment.

4.2 FINITE IMAGE SPACES

Consider a 512×512 monochrome image in which each pixel value has been determined from a random number generator in the range of 0 (black) to 255 (white). Such an image would look like Figure 4-1. This is a *noise image.* The histogram for this image is almost completely flat, showing that the pixel value distribution is uniform, because of the probability distribution of the random number generator. Other distributions are possible, and distributions can be changed on a per pixel basis. If we process the image with a 3×3 mean filter, as shown in Figure 4-2, we see that some of the randomness is gone. The mean filter replaces each pixel with the average value of the pixels that surround it in a 3×3 neighborhood. Because of the averaging effect, pixels have been smeared into larger, homogeneous globs. The image now looks more like a texture, or pattern. The histogram is no longer flat, but Gaussian (or normal) in shape. What we have done, in effect, is to generate Gaussian noise.

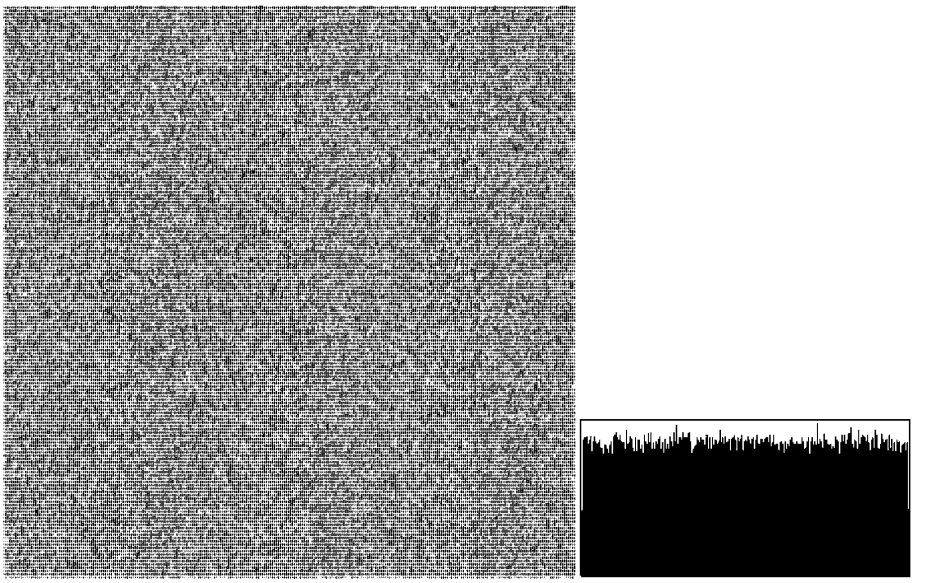


Figure 4-1. Noise image and histogram.

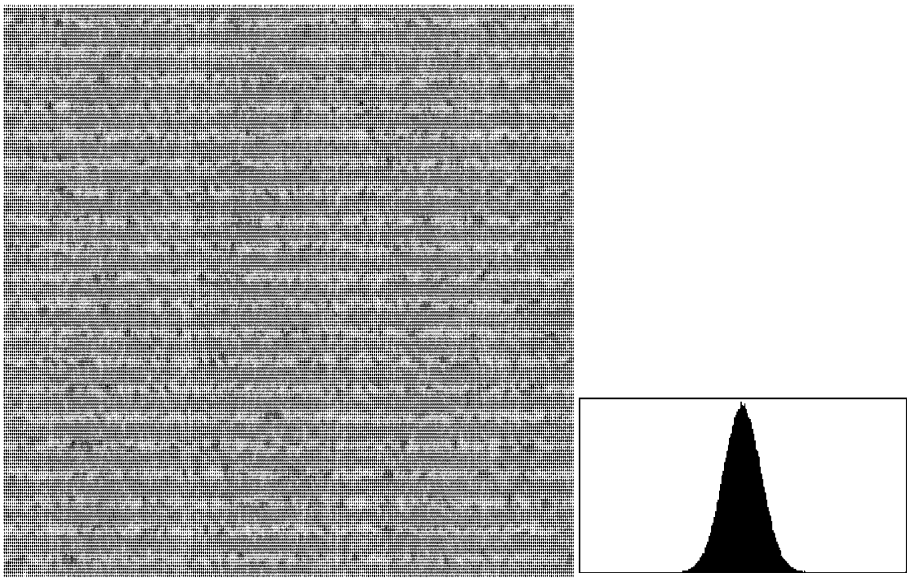
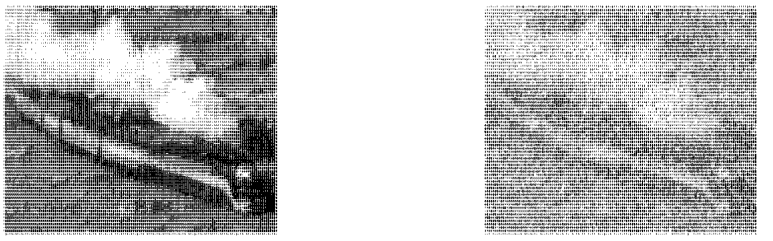


Figure 4-2. Mean filtered noise and histogram.

Now examine Figure 4-3(a) and 4-3(b). In (a) there is a grayscale picture of a locomotive rounding a curve and (b) the same scene is shown with noise overlayed. The noise image is similar to that of a noisy television channel if we were to capture a single frame from it. The point is that once a sampling and a quantization have been selected for an image set, that set is constrained and will contain a finite number of images, although that number is huge. The resolution 512×512 was chosen because it approximates the resolution of a television picture. Imagine every image you have ever seen on a black and white television—each of those images is within the set just described. The membership of images in the set is a permutation of 262,244 pixels taken 256 different ways. The total number of images is then immense, on the order of 262,244 factorial.



(a) (b)

Figure 4-3. (a) Locomotive, (b) with noise added.

In general, the less randomness an image has, the more likely the image will be recognizable as a natural scene. In this sense, machine vision can be thought of in terms of a search through the space of all images within a set as described above. Think of the last movie that you saw in black and white. Every frame of that movie is contained within the set of $512 \times 512 \times 8$-bit monochrome images. A standard movie lasts about two hours. At 16 frames per second, that comes to a little over 100,000 frames. If we took a vision system and put a random number generator on every pixel of the image display so that all of the images possible in the set would eventually appear, you would see every frame in your movie...eventually. The idea is to reduce the set by constraining it in some way that is pertinent to the application. Part of the constraining can be in knowing what you are looking for in the vast ensemble of images in the set. Another way to constrain the set is to control the environment in which the set is acquired. This is particularly easy to do in a factory environment where lighting, speed of conveyor systems, and the identity of the objects being looked for is known.

In some situations, we must enlarge the set in order to include an object or feature. For example, if the camera chosen for a task cannot resolve an object of interest, such as a tumor on an X-ray photograph, then the resolution must be increased and the set of possible images enlarged. In this case the enlargement of the space may be offset by having a specific region in which to look. With a mammogram, for instance, the system would only look in the central portion of the image. We conclude this section by noting that it is useful to think of images in terms of sets of possible scenes based on the constraints established by the application or task to be performed. We show this in greater detail in the next section.

4.3 APPLICATIONS

Although many applications for machine vision exist, it will be useful to review the limits of the field. At the lower end is the problem of the conveyor belt in the factory. There, a camera images parts and a vision system is used to reject a part or accept it, which was shown in Chapter 3. At the outer reaches of the field there is a human-level vision process driving a robot. At the low end, the conveyor belt problem has been solved and machine vision systems are in place in many factories and industrial sites. The conveyor belt problem requires pattern recognition, object registration, alignment, and, in some cases, tracking. Algorithms to perform these functions are well known and available in commercial packages. For now we will examine a number of machine vision packages that are in place and have been tested in the field. This will complete the chapter. In Chapter 7 the discussion will be on how to process objects and regions and we will save the discussion of human-level machine vision for Chapter 8.

4.3.1 IDENTIFICATION AND SORTING OF FISH

A system for the automatic identification of fish species for shipboard processing was developed by Scotland's Torry Research Station in Granite City, Scotland.[2] Netted fish enter a conveyor belt system and are imaged in 24-bit RGB. Silhouettes of the fish are also imaged by a fluorescent light box under the transparent conveyor belt. After image acquisition and processing, the fish proceed to a series of diverters (pneumatic arms that direct the fish into sort bins) that are controlled by the vision system.

The vision processing first determines the orientation of the fish. Fish appear randomly oriented on the conveyor, although the feed system guarantees that only one fish will be present in an acquired image. A color thresholding (Section 5.1) and chain linking (Section 6.1) algorithm are used to develop an outline of the fish shape. First-order moments (Section 5.3) are then employed to determine the principal axis of the fish. The width of the fish is then calculated at 10% of overall length from each end and the wider end is identified as the head, establishing orientation.

The individual species of fish are determined by using a template matching scheme (Section 6.2) and mensuration (Section 5.3). In template matching, the first-order moments are computed and aspect and area ratios are calculated. These ratios are then compared to standard templates of the species known to the system. The mensuration scheme determines the width of the fish at equal distances along their length and compares this data to standard distributions. This scheme is illustrated by Figure 4-4. The vision system would determine the fish widths at each of the positions W_1 through W_4. The widths determined are then compared against a database of average widths at these points from various species. The closest match is then judged to be from that species.

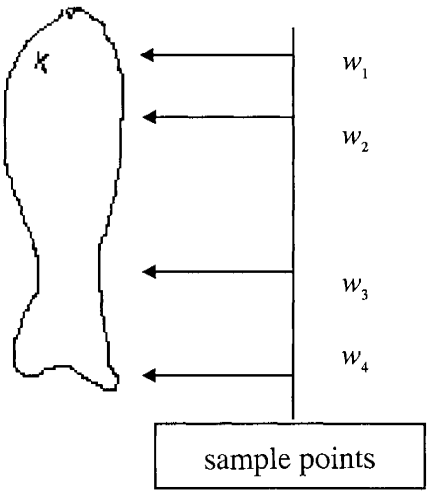


Figure 4-4. Fish identification system width measurements.

In both approaches, when a fish is curled in the image, both the ratio and width measurements produce inaccurate results as they assume that the fish is lying flat. To work around this, normals are computed from the main axis of each fish species to the edge of the body to allow detection of a curled fish, which might be identified incorrectly using the previous methods. The width feature is made invariant by this step and the problem of curling is minimized.

As a final processing step, the color and shape of the fish are compared to a database as a final discriminant. The system has been shown to be 99% accurate in sorting at a rate of 60 fish per minute. It should be clear that this system uses an ensemble of methods to achieve the goal of identifying fish species. Each processing step is given a vote in the final decision of which species the fish is. The determination of how much a vote should count is often based on heuristics, or rules-of-thumb. Heuristics are largely determined from experience or common sense and are sometimes difficult to express empirically. In the case of the fish, the color attribute may have a stronger vote in the case of a fish that is brightly and uniformly colored. It would contribute less to a decision between two plain-colored species.

4.3.2 OBJECT COUNTING

The counting of randomly-scattered objects on a conveyor system can be considered as an exercise in constraining the *geometry* of the machine vision problem, as we discussed in Section 3.3. The Sci-Agra Company of Fort Wayne, Indiana, has developed one such approach.[3] Their census system uses a linear infrared (IR) emitter/detector array to develop a 3-D representation of what passes beneath it on a conveyor belt, as illustrated in Figure 4-5. The intensity of the reflected IR beam is proportional to the distance of the object from the emitter/detector pair.

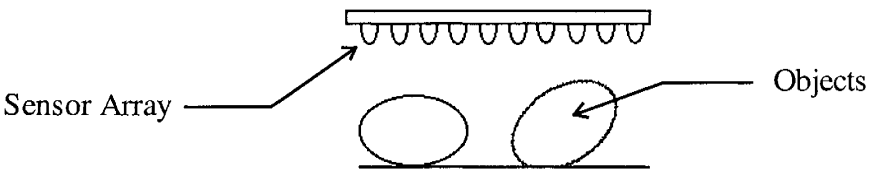
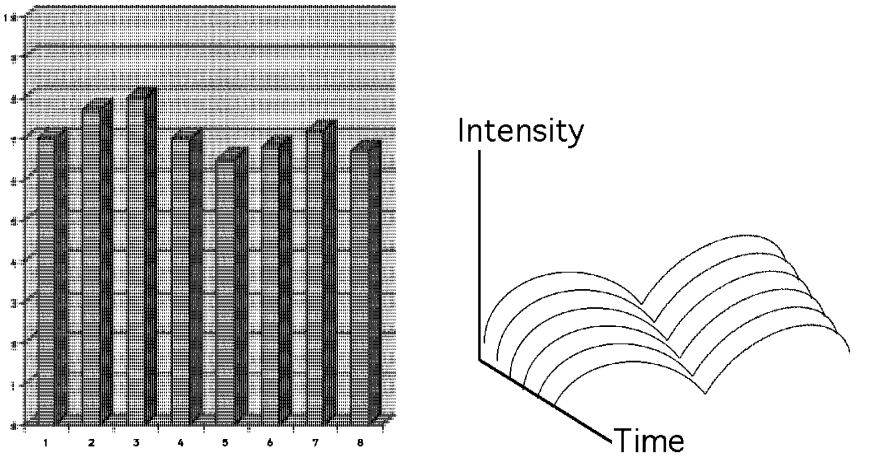


Figure 4-5. Census IR counting system.

The output of the array at any moment in time is illustrated by the plot in Figure 4-6(a). As the conveyor moves and the array is sampled, a 3-D representation is created, Figure 4-6(b). Custom software is developed depending on the complexity of the object or objects that must be counted. The system has been used successfully on eggs, baked goods, and frozen foods. The primary algorithm used is surface-area calculation (Section 6.2).



(a) (b)

Figure 4-6. (a) Sensor output. Vertical axis is reflected IR intensity, while horizontal axis is position; (b) 3-D representation showing multiple sensor outputs in time-revealing egg shapes.

4.3.3 VEHICLE LICENSE PLATE NUMBER SENSING

The last application is that of automated detection and evaluation of vehicle license plate numbers. A system was developed by the Perceptics Corporation of Knoxville, Tennessee, to locate and read the license plate numbers of vehicles entering Canada at the Rainbow or Whirlpool bridge border crossings in Niagra Falls, Ontario.[4] When a vehicle is sensed at the crossing point by breaking an infrared (IR) beam, a flash of near-IR light is fired in conjunction with the electronic shutter of an IR CCD camera aimed at the rear of the vehicle. The image is digitized and the license number evaluated in a matter of seconds.

The first stage of the processing is to resolve the location of the license plate from numerous other rectangular objects in the image such as bumper stickers, manufacturer logos, parking permits and the like. Location of the plate is determined using a decision tree structure that applies a number of different tests to the image. These tests are proprietary to Perceptics, but one can surmise that they consist primarily of edge detection and shape locating. Once the plate is located, the characters on the plate are separated from the background and analyzed using a *structural analysis* approach.

Structural analysis requires that the geometry of the target object is determined, in this case letters and numbers. This is distinct from template matching approaches that try to correlate unknown objects that have been extracted from an image to a database of known objects. The domain of the identification is somewhat restricted as the system knows information such as fonts and syntax for Canadian and U.S. plates, which make up the bulk of the

plates that the system sees. Because the domain is restricted and the evaluation is constrained to the area of the plate handed off from the initial processing step, the system is very robust and highly accurate. Exact figures are not known, but the Canadian government has equipped most of its 120 land-based ports of entry with the system. In Section 6.2 we will examine representation methods that include structural approaches used in this system.

REFERENCES

1. S. Zeki, *A Vision of the Brain*, Blackwell Scientific Publications, Oxford (1993).

2. J. Shave, "A shipboard imaging system to sort and identify live fish," *Advanced Imaging*, pp. 57-59, May 1993.

3. T. Blomenberg, "Breaking new ground in machine vision," *Advanced Imaging*, pp. 28-30, August 1994.

4. L. J. Nelson, "ICR-based automatic license plate reading for Canadian customs," *Advanced Imaging*, pp. 29-30, March 1993.

CHAPTER 5
OBJECTS AND REGIONS

In Chapter 4 we studied the problem of machine vision and concluded the chapter with a brief look at three interesting applications. With this chapter we begin to examine the fundamental methods of machine vision processing—the breakdown of an image into objects and regions.

5.1 THRESHOLDING

Thresholding is the separation of objects or regions in an image based on pixel graylevels above or below a selected (threshold) value. A simple mathematical definition of thresholding is given by

$$g(x,y)=\begin{cases} G_0 \; if & f(x,y)>T \\ G_B \; if & f(x,y)<T \end{cases}$$

where G_0 is the replacement value for the object, G_B is the replacement value for the background and T is the threshold value. This is best illustrated by an example. Consider Figure 5-1, the image of a cruise ship and Figure 5-2, the histogram of this image. If we let T=200 (as indicated on the histogram), G_0 =255 and G_B=0, the result is illustrated in Figure 5-3.

Figure 5-3 reveals an object, a cruise ship, that is the result of simple thresholding. This tells us that the small peak at the far right of the histogram is comprised of pixel values from the ship object. We might speculate that the large peak represents sky pixels. In this case, we can employ a technique known as *density slicing* or multiple thresholds. We can replace all pixels below value 200 and above value 160 (the range of values of the large peak in the histogram) with 255, the object value, and all other pixels with zero, the background value. This operation yields the region of sky as shown in Figure 5-4.

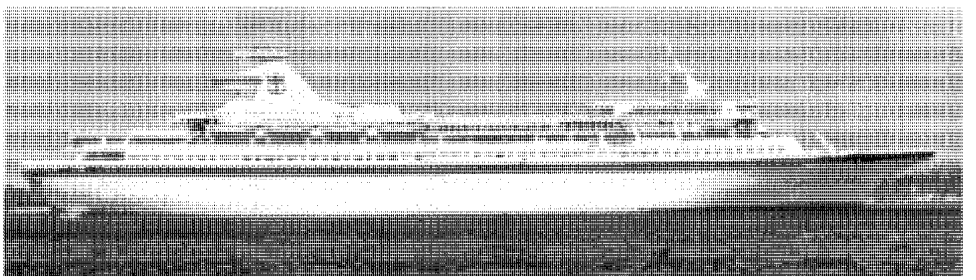


Figure 5-1. Cruise ship.

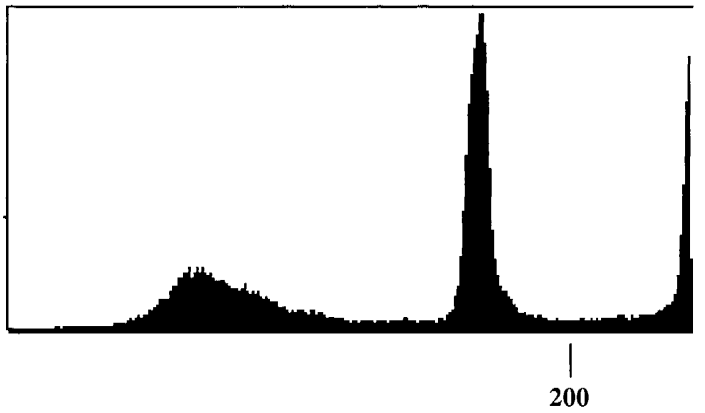


Figure 5-2. Cruise ship histogram.

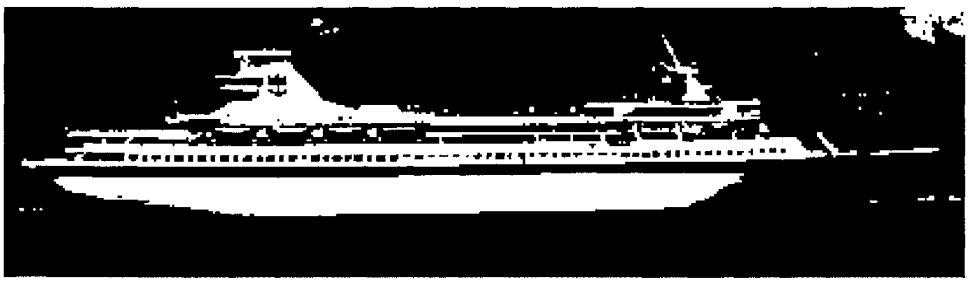


Figure 5-3. Cruise ship thresholded to level 64.

Figure 5-4. Density slice of cruise ship yielding sky region.

The question that must now be answered is, "How do we choose the threshold values?" This is a difficult problem and a great deal of research has been undertaken to find automated methods of thresholding. We will now look at two techniques of automated thresholding: optimum and class variance.

5.1.1 Optimum Thresholding

Optimum thresholding assumes a bimodal histogram and seeks the minima as the threshold point. For example, consider Figure 5-5(a), an image of small parts, and its histogram, 5-5(b). The parts in the image, from left to right, are: a small toy pot, open-end down; the metal clip from a key chain, the clip is open; a Hall Effect Gear Tooth Sensor, this item is black plastic and has a cylindrical shape, one end has a mounting tab where the connection wires are attached (these are visible in the picture); and lastly, a large paper clip. This image was used for a project in a machine perception course that the author teaches. It is a good example of an image that might be encountered in machine vision problems where objects must be located and identified. A histogram is almost always the first data extraction that is made on an image prior to further processing (see Section 2.5, Point Processes). The histogram can reveal a great deal about the objects in an image, but the interpretation of a histogram is not always straightforward or useful. Note that the histogram for the parts image is bimodal, meaning that it has two distinct peaks.

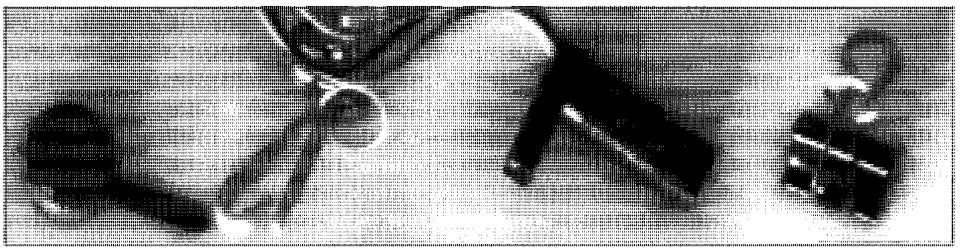


Figure 5-5. (a) Image of small parts.

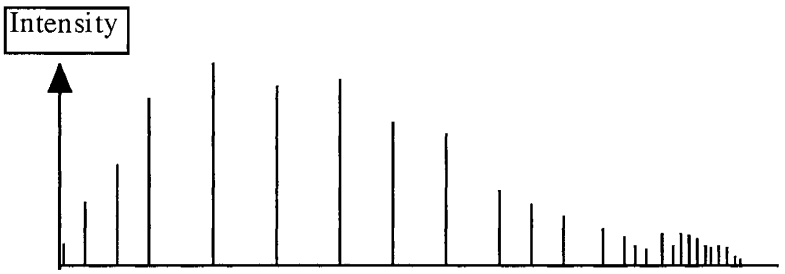
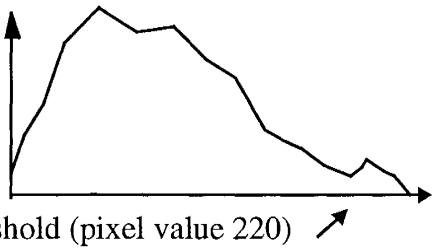


Figure 5-5. (b) histogram.

If we convert the histogram into a piecewise-linear function, and then seek the transition point of the modes, we get the following:



We call this transition point the "optimal threshold," which in this case turns out to be 220. This is the threshold where the bimodal distributions meet. If we threshold the small parts image [Figure 5-5(a)] to this level, the result will be as shown in Figure 5-6. The algorithm that performs optimal thresholding is the same as the algorithm for determining the minima of a function while excluding the end points. The algorithm assumes that the histogram is bimodal and that the *objects of interest lie to one side of the minima*. The underlying assumption here is that the background pixels (pixels not belonging to the objects of interest) will be uniform and make up the bulk of the image pixels. Generally, this can be controlled through proper lighting of the workspace. In situations where the system encounters shadows and variable lighting conditions, this approach fails and more sophisticated techniques must be employed.

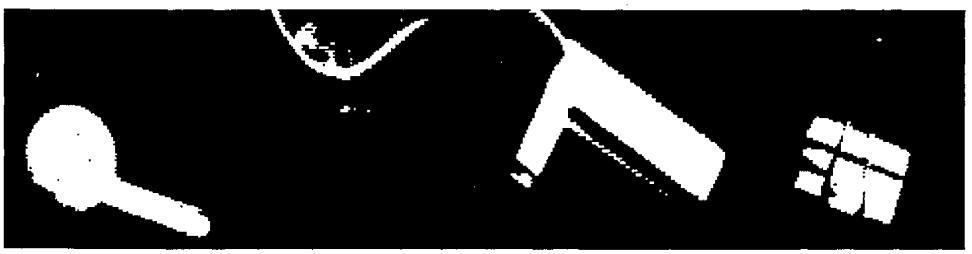


Figure 5-6. Optimal thresholding (level 220) of Figure 5-5(a).

5.1.2 Class Variance Thresholding

Class variance thresholding was first proposed by Otsu and is sometimes called Otsu's method.[1] The algorithm maximizes the separation of classes using a discriminant criterion based on zero and first-order moments of the histogram. The method is nontrivial and is best explained in terms of the expression used to evaluate it, the between-class variance. This is given by the following expression:

$$\sigma_B^2(K) = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]}.$$

The components of the between-class variance are as follows:

Mean value of image $$\mu_T = \mu(L) = \sum_{i=1}^{L} ip_i$$

First-order cumulative moment $$\mu(k) = \sum_{i=1}^{k} ip_i$$

Zeroth-order cumulative moment $$\omega(k) = \sum_{i=1}^{k} p_i ,$$

where L is the number of graylevels in the image and p_i is the number of pixels at graylevel i. These equations may appear intimidating, but the simple explanation of the moments is that they provide a scalar value that summarizes the relationship between pixel values and spatial position. This makes sense heuristically when you consider that we are attempting to find a threshold that yields the best separation between objects in the image.

In Otsu's method, the optimal threshold k^* is the graylevel when the between-class variance is maximized. This will occur (theoretically) when the squared error developed between the image mean scaled by the zeroth- order moment (a measure of dispersion of the histogram) and the first- order moment (a measure of dispersion weighted by the grayscale value) is large. This means that a threshold that divides the pixel values into maximally-unequal portions will evaluate to the maximum between-class variance. The idea is that objects will be minimally dispersed in grayscale whereas background will be maximally dispersed. We can express the computation of the optimal threshold, k^*, as:

$$\sigma_B^2(k^*) = \max_{1 \le k \le L} \sigma_B^2(k).$$

An example of Otsu's method is shown in Figures 5-7, 5-8 and 5-9(a) and (b).

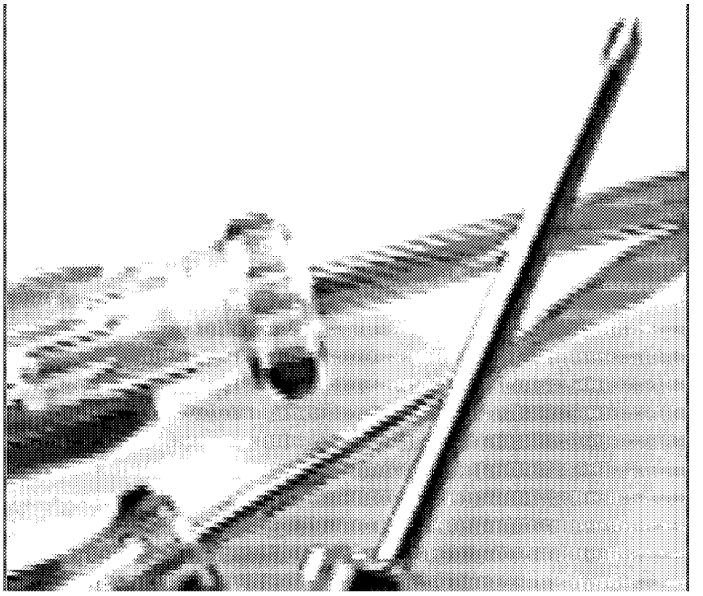


Figure 5-7. Screwdrivers.

Examination of Figure 5-8, the histogram of the screwdrivers image, indicates that Otsu's method found a prominent minima in the histogram that adequately thresholded the key features of the screwdrivers from the background. If we used the bimodal thresholding algorithm of Section 5.1.1, it would have selected a threshold in the neighborhood of 190. The results of applying these thresholds to the screwdrivers image are shown in Figure 5-9. Figure 5-9(a) shows the Otsu threshold of 170 and Figure 5-9(b) shows the bimodal of 190. The Otsu threshold removed more of the background than the bimodal, which would simplify the automated recognition of the screwdrivers.

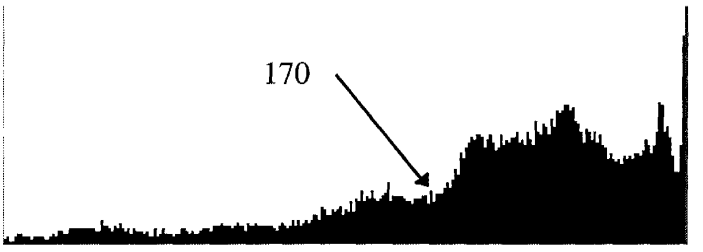


Figure 5-8. Histogram of screwdrivers image (Figure 5-7) with notation of threshold from application of Otsu's class variance thresholding.

a

b

Figure 5-9. (a) Results of Otsu's method applied to screwdrivers image, optimal threshold of 170. (b) Results of using a bimodal threshold of 190.

5.2 SEGMENTATION

Segmentation is the process of separating objects and regions from images. Although we did not mention it earlier, thresholding is the simplest form of segmentation. It should be clear at this point that the fundamental problem in thresholding is the selection of a threshold that will reveal the best segmentation. Since segmentation is defined in terms of objects and regions, the accurate separation of these elements from an image is often used as a measure of the success of a segmentation scheme. Segmentation can get very complex and involves a sequence of steps. For our purposes, we will tie segmentation to thresholding by considering segmentation as a feedback process, as illustrated by the diagram in Figure 5-10. The input image, $f(x,y)$, is thresholded to threshold T and evaluated yielding a segmentation $g(x,y)$.

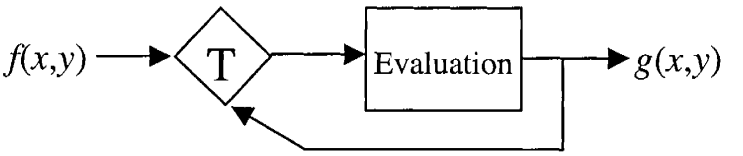


Figure 5-10. Segmentation process.

If the evaluation is unsatisfactory or insufficient in terms of number of objects revealed or identified, the threshold is adjusted. Multiple thresholds may be necessary in order to resolve all of the objects and regions from the image, hence the result $g(x,y)$ follows the evaluation process.

As an example, the evaluation process might be the segmentation of a single object. The segmentation would proceed with various thresholds up through the grayscale until the evaluation process detected a single object. This is shown in Figure 5-11, where the first image (a) is that of a pair of pliers, 256 graylevels; the second image (b) is the pliers image thresholded to level 150; while the third image (c) is the threshold at 187. All of the white pixels in Figure 5-11(b) belong to the pliers object, while those in (c) belong to a number of independent regions. All of the pixels in (c) are part of the pliers, but they are fragmented.

The algorithm to do this detection is simple. At every threshold we determine whether or not each pixel above the threshold has a neighbor. If this test passes, then we have a single object. One might argue that the threshold in Figure 5-11(b) is probably not unique, and that other thresholds may exist that also reveal the pliers as a single object. We can add a test to require that the final threshold reveals the single object *with the maximum number of contiguous pixels*. A problem that can arise is that at some point there might be a threshold that has a contiguous object that contains the pliers and also portions of the background. With this logic, eventually we will come to the

zero threshold, which will be the largest single object in the image because all pixels will be included! To avoid this we can place an upper limit on the number of object pixels, based on *a priori* knowledge about the object. The evaluation process can contain other, more sophisticated, means for setting the threshold. Many of these fall under the category of mensuration, which we discuss in the next section.

(a) (b) (c)

Figure 5-11. (a) pliers, (b) threshold at 150, (c) threshold at 187.

To continue this line of reasoning regarding the setting of a threshold for a single object, we can extend the concept to deal with multiple objects. If we are aware that we have acquired two or more objects, or should have acquired this number, then the evaluation process must be increased in sophistication. To determine multiple objects in a thresholded image, the algorithm can mark pixels that are members of unique objects by setting the thresholded pixels to specific grayscale vales. An example of this is illustrated in Figure 5-12. Here, the first image is that of a hammer and a pair of pliers. The second image is after thresholding. The square area near the hammer head and the tips of the pliers has been expanded to show detail. The pixels from the hammer are set to one gray value while the pixels from the pliers are set to another. In later operations on the image, this distribution "marks" the coordinates of object pixels. This marking operation can be performed by a clustering algorithm, one of which is discussed in detail in Section 7.2.

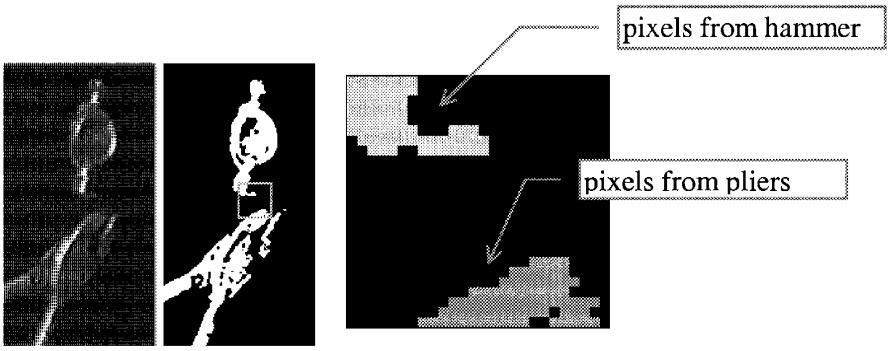


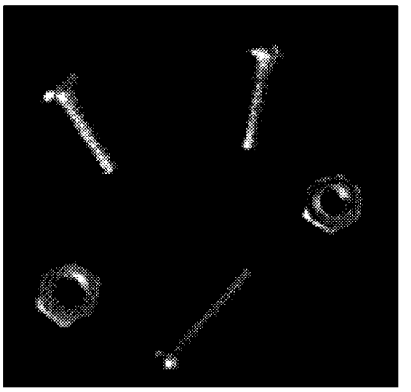
Figure 5-12. Multiple object marking.

5.3 MENSURATION

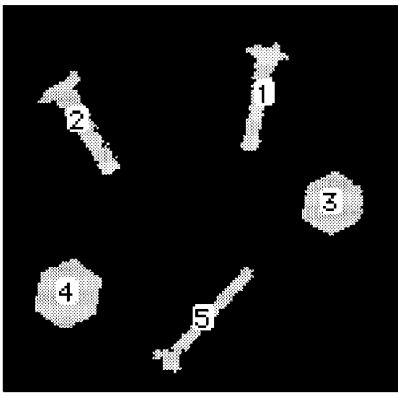
Mensuration is the act or process of measuring. In machine vision, we are interested in making measurements of image objects and regions for the purpose of segmentation and recognition. Once objects have been segmented, measurements are taken. These measurements may then be used in recognition algorithms to identify the object or region. The segmented image provides a template, or coordinate list, of thresholded objects. From this list of coordinates, the pixel values from the original image can be analyzed for statistical features such as density and standard deviation, or minimum and maximum values. The number of pixels in an object represents the area of the object. If the pixels have been calibrated to actual distance measure, an accurate assessment of actual object size can be determined. The pixel coordinates can be processed for centroid (or x-y center), perimeter length, and major and minor axes. Other measurements are possible, as well as combinations of measurements. An example of measurement is shown in Figure 5-13. In this example, an image of nuts and bolts against a black background has been analyzed, 5-13(a). The image has been thresholded and the objects located and identified by numbers, 5-13(b). The table at the bottom of the figure, 5-13(c), shows the area of each object and the mean.

The area values are computed by counting the number of pixels in each object that exceeds the threshold. The mean is evaluated from the set of original pixel values that exceed the threshold. From the table, we can state that "If an object has an area greater than 300, then it is a nut, else it is a bolt." We can also state that "If the mean value of the object is greater than 40 and less than 50, then the object is a nut, else it is a bolt." A potential problem with this reasoning is that the measurements are dependent on the threshold. Consider Figure 5-14(a), the thresholded objects from Figure 5-13(b), and compare this to 5-14(b), an alternate threshold. The area values for the objects will be different and it is unlikely that we could apply the same reasoning from before to the identification of the objects. In other words, an area greater than 300 might now be a bolt instead of a nut.

If the objects are not positioned consistently, for example, if the nut object was on end instead of lying flat, measurement will be affected. In the industrial environment, positioning can be controlled by using vibration tables, variable-width transport channels, and other devices to control and guarantee object presentation to the imaging system.



(a)



(b)

Object	Area	Mean
1. (bolt)	264	66.79
2. (bolt)	289	70.66
3. (nut)	319	47.13
4. (nut)	376	49.22
5. (bolt)	237	33.44

(c)

Figure 5-13. (a) nuts and bolts, (b) thresholded objects, (c) table of area and mean measurements on objects.

(a)

(b)

Figure 5-14. (a) threshold for 5-13(b), (b) alternate threshold.

The sensitivity of mensuration to threshold cannot be underestimated. To achieve consistent thresholding, the lighting and background variables are the most important. Once again, in the industrial environment, the control of these elements is relatively easy. Lighting should be selected to have an emission wavelength consistent with the camera sensitivity. Intensity should be adjusted to minimize or eliminate shadows. The background should enhance the contrast of the objects as much as possible.

To conclude this section and chapter, we give a table of measurements and their descriptions. The same reasoning that was applied in Figure 5-13 for area and mean can also be used with all of the measures listed. There is a direct correlation between the robustness of an object recognition system and the number of measures it employs. However, there is a corresponding increase in the computational resources needed by the machine.

This is by no means an exhaustive list. A major facet of machine vision research is the discovery of new features of objects that can be evaluated from an image for the purpose of object recognition. Specific algorithms, examples, and computer code for the measures listed in the table can be found in Ref. 2 below.

Measure	**Description**
Area	Count of pixels in object.
Mean	Mean value (average pixel value)
Variance	Variance of pixel values—measure of deviation from the average.
Standard Deviation	Standard deviation of pixel values—measure of dispersion, root of variance.
Centroid	Coordinates of the center of pixel "mass." Identifies the location of the object in the image.
Modal Density	Most frequently-occurring pixel value in the object.
Perimeter	Number of pixels in the perimeter or outline of the object. For a circle object, this would correspond to the circumference.
Maximum Axis	The axis of minimum dispersion passing through the centroid.
Measure	Description.
Minimum Axis	The axis of maximum dispersion passing through the centroid.
Angle	The angle of the maximum axis with the horizontal (x) axis of the image.
Min/Max	Minimum and maximum pixel value of the objects.

REFERENCES

1. N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," IEEE Trans. SMC-9:1, January 1979.

2. H. Myler and A. Weeks, *The Pocket Handbook of Image Processing Algorithms in C*, Prentice-Hall, Englewood Cliffs, 1993.

CHAPTER 6
RECOGNITION

Recognition is the core of machine vision. Algorithms are used to determine the identity of objects and regions that are segmented from scenes. The discussion of mensuration in Section 5.3 was a prelude to object recognition and it explained how measures could be used to discriminate between objects in a scene. In some sense mensuration belongs with recognition, but in this chapter we will explore the more advanced concepts of representation and feature analysis.

6.1 REPRESENTATION

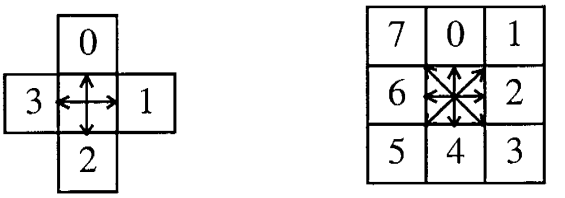
Representation is the process of determining a description for an object or region. Many representation schemes have been proposed over the years and the classical descriptors can be classified as either boundary or regional descriptors. The descriptors to be discussed are:

- boundary: chain coding
- boundary: polygonal approximation (curve fit)
- boundary: signatures
- region: skeleton
- region: topology
- region: texture

Advanced descriptors such as 3-D and surface volumes will also be discussed.

6.1.1 BOUNDARY DESCRIPTOR: CHAIN CODING

Chain codes were first introduced by Freeman[1] and are sometimes called "Freeman chain codes." The chain code is a way of directionally encoding a pixel boundary. A set of directions is employed that can be four, eight or more directions. To visualize this one can think in terms of the compass points NSEW. Two forms of these codes are shown below:



four directions eight directions

To represent a contour, the direction from the first pixel to the second is determined and recorded. Then the direction from the second to the third, and so on. Consider the example shown in Figure 6-1.

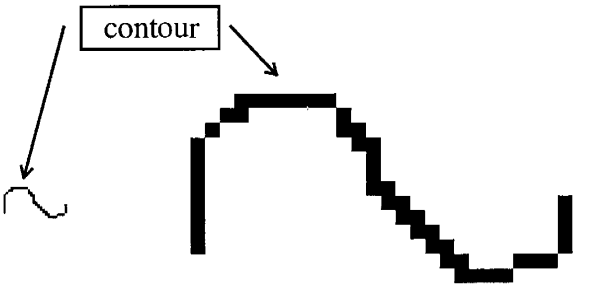


Figure 6-1. Contour magnified to illustrate pixel directions.

The 8-direction chain code for the contour in Figure 6-1 would be:

0000000112022222224424442424242424222022200000

From just looking at the code we can see that this is just a contour and not a closed boundary. In Figure 6-2, we show a simple closed boundary.

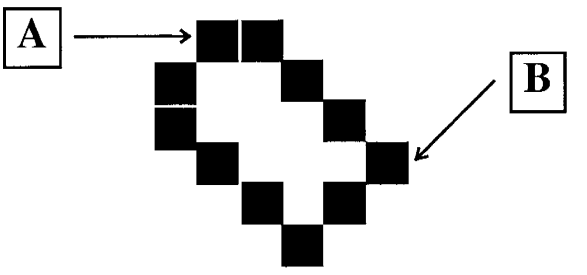


Figure 6-2. Simple closed boundary.

If we start the code at pixel A, then the 8-direction code is:

23335577701

and if we start at pixel B, the code becomes:

557777012333

To select a consistently unique code for a contour, we can compute the smallest integer code. In this case, the code 012333557777 would be used. Whether the contour is closed or not depends on what is being looked for. Once the codes are processed, they must be evaluated according to some criteria that is determined from the application. If the application involves looking for tumors in x-rays, then the contours will most likely be closed and circular (although some tumors are randomly shaped). Likewise, if the application is evaluating microscopic images of chromosomes, the contours will have shapes more like those in Figure 6-1. Chain codes can be computed using sampling grids, an example of which is shown in Figure 6-3. The problem with grids is that they are sensitive to contour rotation, position, and scale. Obviously the finest grid possible is that of the image sampling itself.

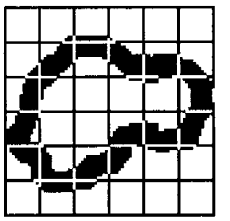


Figure 6-3. Chain code sampling grid.

Problems can arise when contours and boundaries are not smooth or have missing pieces. This problem is called *fragmentation* and many techniques have been employed to overcome the problem. Figure 6-4 shows the contour of Figure 6-2 with fragmentation. The judicious selection of a sampling grid can sometimes overcome the effects of noisy contours. Additionally, the chain code can be treated as a single-dimensional signal and processed using filters, correlation algorithms, and spectral analysis to fill in or smooth the contour.

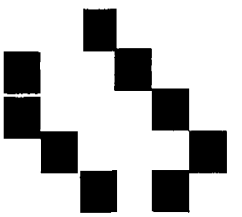


Figure 6-4. Chain code with fragmentation.

The chain code as a representation is useful primarily to confirm other schemes used in a recognition algorithm suite or to compress boundary and contour data. The chain code can adequately represent a complex contour in much less data space than the contour data present in an image. An upper bound on the length of the code (assuming maximum sampling) is the number of pixels in the contour. This is always one half of the number of Cartesian coordinates needed to represent any given contour. To see this, consider the contour under the sample grid in Figure 6-3. Each pixel in the contour requires a set of coordinate pairs to identify it in the grid. Using a chain code, only one datum is required per pixel—the direction. The chain code, however, does not locate the contour in the grid. This can be specified using the centroid of the object producing the contour (Sections 6.3 and 7.2.3).

6.1.2 BOUNDARY DESCRIPTOR: BOUNDARY SPLITTING

Boundaries and contours are treated as points on the Cartesian plane (see Section 6.1.7 for volumetric treatments) so that polygons or curves can be fitted to them. In the simplest case, we can fit a polygon to a boundary using the boundary splitting technique. Here the two farthest pixels are determined and a line is drawn between them. These pixels constitute two initial vertices of the polygon. The two pixels farthest from the bisecting line are then computed. This determines the vertices of a four-sided polygon. The farthest pixel from each of the four sides to the boundary is computed and new vertices are found. This continues until a polygon of the desired order is computed, a minimal distance threshold is reached, or the polygon consists of all pixels in the contour.

Some preprocessing is necessary prior to the splitting sequence and this is illustrated in Figure 6-5. The original object (image at far left of figure) is that of a small fastening nut. It was thresholded, the center pixels removed, and the outline processed. The algorithms needed to process this sequence consist of a thresholder (optimal or class variance would work), an erosion scheme, and an outline process. The thresholding produces a binary image where pixels are either white or black. The center erosion process begins with the determination of the centroid of the thresholded object (Sections 5.3 and 7.2.3). Pixels are then processed from the center pixel out towards the boundary of the object.

Original **Thresholded** **Center-eroded** **Outline**

Figure 6-5. Polygon extraction sequence.

Black pixels are changed to white pixels until no change occurs between two adjacent pixels. The outline is produced by subtracting the center-eroded image from a single-pixel *erosion* of the center-eroded image. Erosion removes one pixel all along the edge of an object while the complementary operation, *dilation*, adds a pixel to the edge of an object. These processes are known as *morphological processes* and further discussion of them is beyond the scope of this book. The interested reader should consult the Dougherty[5] text for an exhaustive treatment of morphology in general and the Myler and Weeks[2] text for the algorithms to perform morphological filtering.

Figure 6-6 shows a set of six figures that illustrates the sequence of splitting the outline figure into an irregular hexagon. The process follows from the upper-left figure, left-to-right, to the lower-right corner. If the problem consists of searching for fastening nuts, then the process can be halted once a hexagon is generated. The hexagon will be well-formed (although irregular in almost all cases) because the process is based on maximal pixel distances. The most difficult part of the process shown here is the preprocessing. If the threshold operation does not reveal the six sides of the fastener, then the process will break down. Setting up consistent lighting conditions can produce a workable threshold.

Boundary splitting is a powerful and easily-implemented representation for closed contours. It is most useful when the objects to be represented have a fixed polygonal shape, such as boxes, pyramids, or the fasteners in our example.

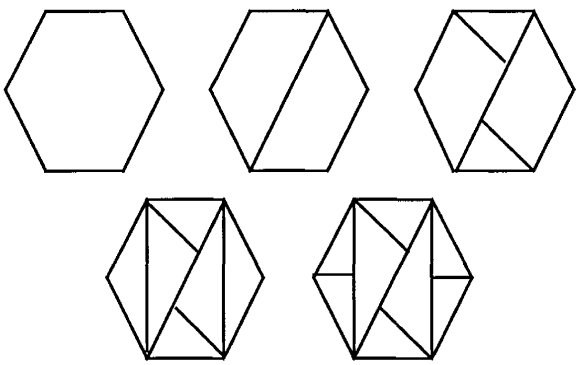


Figure 6-6. Polygon splitting sequence.

6.1.3 BOUNDARY DESCRIPTOR: CURVE FITTING

In the previous section we saw how polygons could be fitted to contours using the splitting technique. Elegantly simple to implement, the splitting technique has the primary disadvantage of generating, in most cases, irregular polygons. Additionally, the representation is not analytic, but consists of a table

of vertices. The analytical result produced by curve fitting may be more useful for later processing because of accuracy considerations. Because of this, for more complex shapes, curve fitting is often the representation of choice. The most popular of curve-fitting algorithms is the ellipse fit. Although computationally intense, the fitting of an ellipse to a cluster of points can lead to unique features for recognition.

A conic section is described by the following equation:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

and the section will be an ellipse if b^2-$4ac < 0$. The section will be a circle if $b^2 = 4ac$. An ellipse can be fitted to a collection of five points by using their coordinates in the equation above and solving the five simultaneous equations for the coefficients. Ellipses are characterized by the lengths of their minor and major axes. The ratio of the axes lengths is called the *circularity*. When this ratio is one, the ellipse is a *circle*.

One can fit an ellipse to an object using an alternate approach to the direct fit method outlined above. Consider Figure 6-7 below for the discussion that follows.

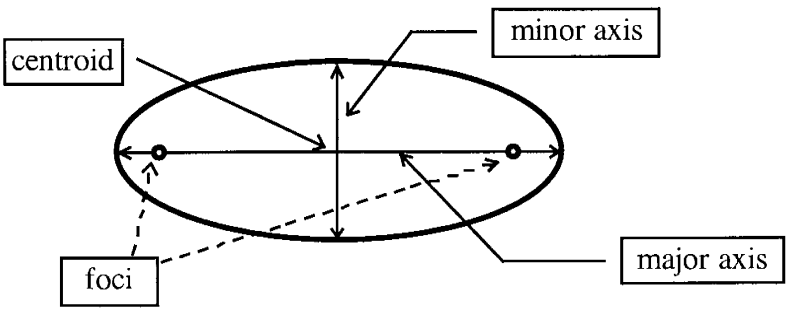


Figure 6-7. Ellipse parameters.

The length of the major axis is given by $2a$, and that of the minor axis by $2b$. The *eccentricity* is given by

$$\varepsilon = \frac{c}{a} = \frac{\sqrt{a^2 - b^2}}{a},$$

where c is the distance from the centroid to the foci, $c = \sqrt{a^2 - b^2}$. The eccentricity, unlike the circularity, will be zero when the circle is an ellipse. The equation of the ellipse in terms of a, b and the centroid (x_0, y_0) is:

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}=1\,.$$

The value of $2a$ and $2b$ (i.e., the lengths of the axes) is easily determined from computation of the object's smallest and largest axes using moments.[2] Figure 6-8 illustrates the ellipse-fitting operation on the thresholded image of the head of a hammer. The curvefitting of an ellipse is far simpler computationally than the polygon fitting discussed in Section 6.1.2. The result is also more compact to store (centroid and ellipse shape parameters).

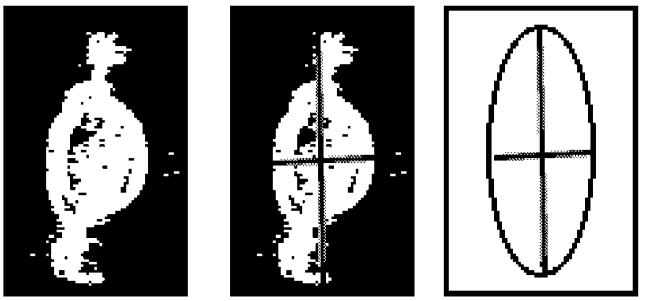


Figure 6-8. Ellipse fitting to thresholded side view of ball-peen hammer head.

6.1.4 BOUNDARY DESCRIPTOR: SIGNATURES

If we process the pixels of a boundary or contour from their two-dimensional coordinates into a single-dimensional representation, we have what is called a *signature*. The chain code can be classified as a signature, however, the term signature is most often applied to closed contours or object boundaries. There are many ways to compute signatures, the simplest being a plot of distance from the centroid of the object to each boundary pixel as a function of angle. Consider the simple example below, Figure 6-9, where the signature is plotted every 45°.

Objects with regular-shaped boundaries and large numbers of pixels will produce signatures that are easily recognizable, as illustrated in Figure 6-10.

Signatures can be analyzed in various ways. The amplitude of the signature is directly proportional to the object size, but the shape is dependent on the contour itself. Hence, all circles will have a flat pixel distance signature, all rectangles will have four peaks, etc. Noise in the contour will affect the signature, but the noise can be easily filtered using single-dimensional signal processing techniques prior to analysis.

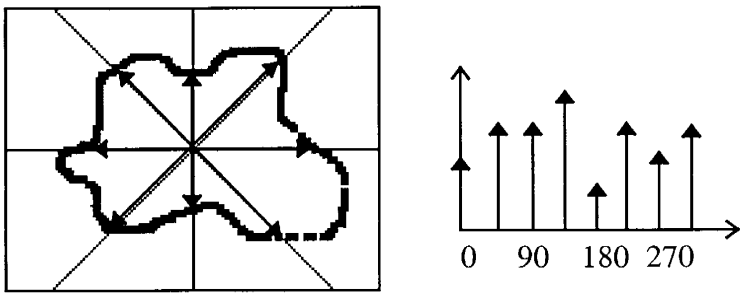


Figure 6-9. Contour and signature.

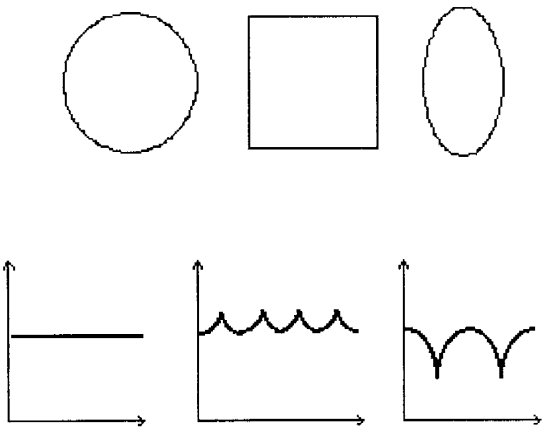


Figure 6-10. (top) Circle, square and ellipse contours; (bottom) signatures.

6.1.5 REGION DESCRIPTOR: TOPOLOGY

Region descriptors require characteristics that reflect the interior features of objects. Topological features are those features unaffected by deformations. For example, one of the simplest of the topological descriptors is the number of holes in the object. Two different objects with the same number of holes are shown in Figure 6-11(a). If we compute the number of connected components, which in both cases above is one, then we can compute the so-called *Euler Number* as:

$$\text{Euler Number} = C - H,$$

where C is the number of connected components and H is the number of holes. Both of the objects above have Euler Number equal to -1. An object with two connected components is shown in Figure 6-11(b). The Euler Number for

this object is 1. Algorithms exist to extract connected components from objects as well as to determine the number of holes.[3] Topological features can also be extracted using polygon splitting and curve fit techniques.

The simplest of algorithms works on binary objects like those shown in Figure 6-11. The Euler number is found by tabulating the coordinates of all pixels in the image and determining connectivity with respect to background. In the case of 6-11(a) and 6-11(b), there would be three classes of pixels in addition to the background pixels: those connected that form the figure (black) and those that form the holes (white). A binary image of an object can be formed using thresholding techniques as discussed in the previous chapter.

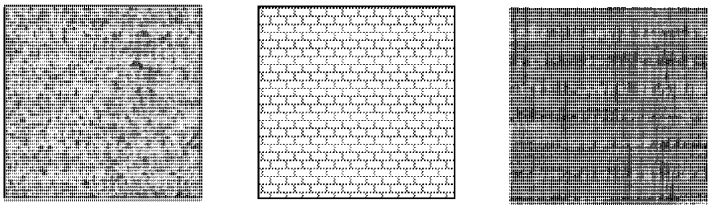
(a)

(b)

Figure 6-11. (a) Objects with two holes, E = -1, (b) object with one hole, E=1.

6.1.6 REGION DESCRIPTOR: TEXTURE

Texture is the property of regions (and objects) that yields a description of their interior characteristics. These characteristics can be defined in terms of repetitive or patterned pixel values, or by the randomness of pixel values. There are three classifications of texture properties: statistical, structural, and spectral. Examples of texture are easy to generate. Figure 6-12 shows three examples of texture: noise, repetitive (brick) graphic, and canvas cloth.

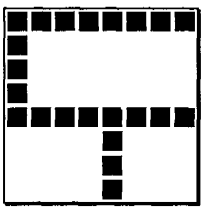


(noise) (brick pattern) (canvas)

Figure 6-12. Textures.

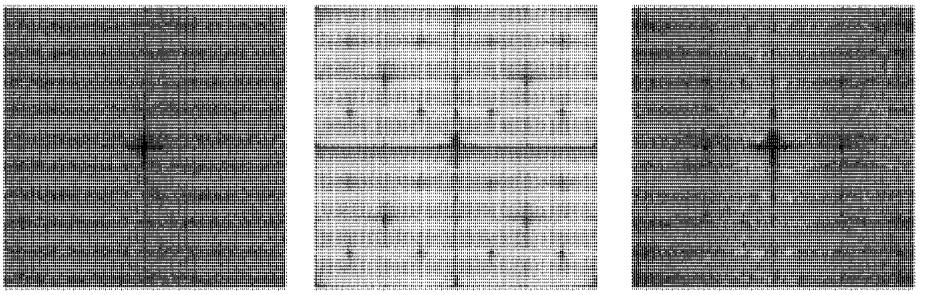
Statistical representations of patterns attempt to describe the aggregate of pixel values using statistics. Statistics are useful in texture analysis if the statistics take into consideration the pixel spatial distribution. It is the spatial distribution that creates the texture. For this reason, second- and higher-order moments are employed. When working with noise textures, the first-order statistics can be very useful as the resulting probability distributions of spatial noise are often well known.

Structural techniques attempt to characterize a "texture primitive," or base element that is repeated to create the texture. This approach works well for repetitive textures like that of the brick wall shown in Figure 6-12. The texture primitive for this texture is given by the pattern below, which is repeated to create the texture pattern.



Determining the texture primitive is a nontrivial task. Among the methods proposed has been the application of production rules to create the texture. This approach has origins in computer language theory. It is easy to generate a set of rules to create a texture, but to determine the rule set that generates a given texture is a problem that has yet to be solved for complex patterns.

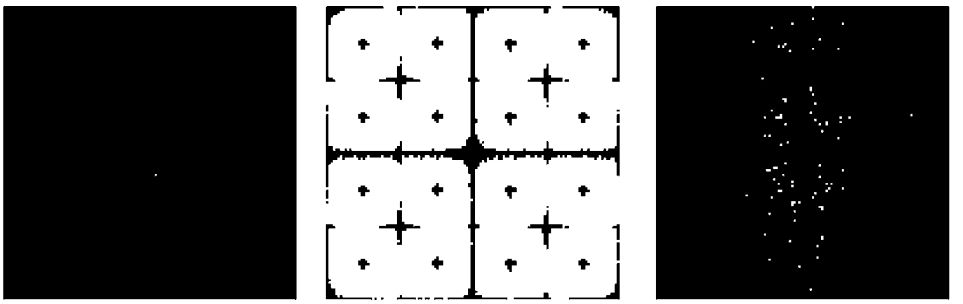
Possibly the most useful and successful of the pattern representation approaches is that of spectral analysis. Here the pattern is analyzed from its Fourier transform. Figure 6-13 shows the transforms of the patterns illustrated in Figure 6-12. Each of the spectra shows distinct patterns, so from the spectra it is possible to easily characterize the texture. The noise spectra is uniform and has no specific features whereas the pattern spectra reflects the pattern in that it has a tile-like repetition that is repeated equally over the entire field. Likewise the canvas, but the repeated tile pattern is more subtle.



(noise) (pattern) (canvas)

Figure 6-13. Fourier transforms (spatial) of Fig. 6.12 textures.

The optimal thresholding of the Fourier transforms in Figure 6-13 yields the binary images shown in Figure 6-14. These thresholded patterns will occur regardless of the size of the object (with the pattern). Discriminating between these can be as simple as assigning the "pattern" label to the thresholded transform image with the most white pixels and the "noise" label to the transform with the fewest. The one left must be the "canvas."



(noise) (pattern) (canvas)

Figure 6-14. Optimal thresholding of Fourier transforms of Fig. 6.12 textures.

6.1.7 VOLUME DESCRIPTORS

Volumetric descriptors are very similar to the curve-fitting boundary descriptors discussed earlier except that the object is a three-dimensional object in three dimensions (3-D). There are two fundamental approaches to this type of description. The first approach considers the intensity of the image to be the third dimension with the pixel spatial position coordinates being the first two. The foundation of this approach has been illustrated in Figure 6-15. The graphic on the left was used in Section 6.1.5 during the topology discussions

while the 3-D view on the right considers a black pixel to be high in vertical distance and white pixels to be low to form a volume plot. Transitions between levels (there are only two in this example) have been enhanced by a gradual grayscale transition to give a smoother 3-D effect. Of course, the goal of this approach is to determine the analytic expression for the volume of an object. There are algorithms to perform this function on unknown objects but they are beyond the scope of this text.[4] Most high-end image processing software programs include the functionality to produce the 3-D plot shown in Figure 6-15. This type of processing can be useful early on in selecting approaches to use in the analysis of the problem.

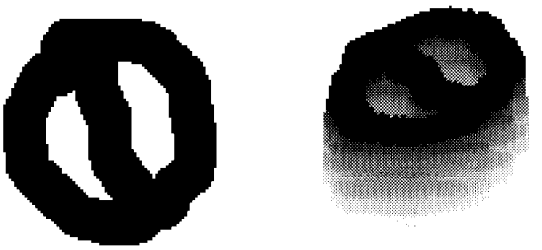


Figure 6-15. Volumetric plot.

The second approach to volumetric descriptors looks at objects as three-dimensional entities. For example, consider the images in Figure 6-16. Figure 6-16(a) is a 3-D rendered graphic of a water tumbler, while (b) is the edge-detected version. In both cases we have a slightly tapered cylinder that is viewed in perspective and exhibits volume properties in spite of the fact that the representation is 2-D. Both images can be stored in two ways: (1) as images, where pixel values represent grayscale, and (2) as graphic objects. As a graphic object, Figure 6-16(b) is called a *wireframe* representation of a tapered cylinder. The computer stores the wireframe object as a set of connected primitives. Most graphics programs include a cylinder primitive, so all that need be stored is the height and diameters of the ends.

(a) (b)

Figure 6-16. Water tumblers.

Figure 6-16(a) is the wireframe object after color and shading have been applied. Both processes, the derivation of a wireframe representation for a real object and the coloring, or *rendering*, of a wireframe object are exceedingly difficult and the exact techniques are far beyond the scope of this text. As discussed in Chapter 3, the usefulness of computer graphics techniques to machine vision cannot be underestimated. Further, large-scale vision systems that incorporate algorithms for volumetric representation are discussed in Chapter 8, Vision Systems.

6.2 PATTERN AND FEATURE ANALYSIS

Once an adequate representation has been achieved, pattern and feature analysis are employed to analyze the image for information content. From our earlier discussion, it should be clear that representation methods are nothing more than feature extraction techniques. We can distinguish between pattern and feature analysis by defining terms:

pattern analysis—a process of defining unique sets of features, called patterns, from a data set whose features or patterns are unknown.

feature analysis—a process where known features are looked for within a data set for the purpose of recognition and identification.

To distinguish between what is considered a pattern and what is a feature, think of patterns as sets of features. The simplest and most fundamental of pattern analysis techniques is cluster analysis, which is the process of counting and labeling objects and/or regions within an image. Clusters consist of pixel groupings that are related to one another by some predetermined measure. For example, in an image of solid-colored marbles, all red pixels could be a cluster, all blue, etc. Furthermore, the clusters could be pixels of a particular color *and* pixels of that color that are adjacent. This would cluster pixels by marble and permit multiple marbles of the same color to be in separate clusters. Clearly, the clustering measure can be just pixel values or a complex set of identifiers or features. A simple clustering algorithm is given below:

Simple Clustering Algorithm

- Specify a size in pixels that the cluster should be.
- Assuming a binary image, use a raster scan to locate the first image pixel.
- Mark the pixel as belonging to cluster number 1.
- Find the next pixel adjacent to the marked pixel, mark it as belonging to cluster 1.
- When no adjacent pixel is found or the cluster size is reached, begin searching for pixels in cluster 2, etc.

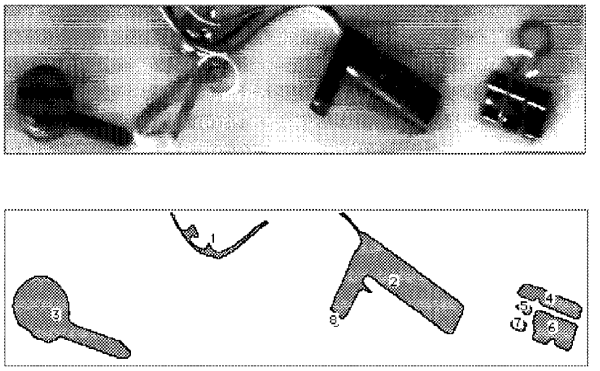
The implementation of a clustering algorithm for grayscale images is not difficult. As the algorithm scans the image, pixel by pixel, it marks membership of a pixel in a cluster by changing the pixels value to the cluster number. For an image with 256 levels of gray, a cluster analysis program based on the algorithm above can identify 255 clusters (plus a background). The clustered image can then be displayed using color to highlight the different clusters found. Figure 6-17 shows the results of running a clustering algorithm on the thresholded image of a set of small objects. This image was introduced in Section 5.1.1 as Figure 5-5. Recall that the parts in the image, from left to right, are:

1. a small toy pot, open-end down,
2. the metal clip from a key chain, the clip is open,
3. a Hall Effect Gear Tooth Sensor, this item is black plastic and has a cylindrical shape, one end has a mounting tab where the connection wires are attached (these are visible in the picture) and,
4. a large paper clip.

The original image is at the top and the thresholded image that was cluster analyzed is below it followed by the results of a feature analysis.

The clustering algorithm employed for the example of Figure 6-17 marks each cluster found with a small number. Cluster number 3 corresponds to the pan or club-shaped object at the far left of the original image. Clusters 1, 2 and 8 are part of the cylindrical black object (a Hall Effect sensor) at the center, while clusters 4, 5, 6 and 7 are part of the paper clip to the far right. Note that the key chain clip did not make it past the threshold operation.

Below Figure 6-17 are the tabulated results of a mensuration performed on the clusters as image objects. The features extracted were cluster area (a count of the number of pixels in the cluster), centroid of the cluster in X-Y coordinates, and length of the cluster (measured on the maximal axis). These features can now be used for a feature analysis to identify which real objects the clusters represent. Assume that these objects are moving on an assembly line. We control lighting (shadows are consistent or eliminated), exposure time (image contrast is constant), threshold (the same objects appear in each frame) and magnification (the size of the objects is constant). From the table we can establish that a cluster with area >3000 pixels will most likely be the Hall Effect sensor. We certainly could add other features to the table to try to distinguish the other objects as well. Another term widely used for clustering is region growing. To avoid confusion, understand that clustering originated from pattern analysis and region growing was coined by image processing researchers. Clustering is highly dependent on threshold. Figure 6-18 illustrates two different thresholds on the same image. Notice the difference in number of objects discernible.



Cluster	Area	X	Y	Length
1.	486	214	118	237
2.	3353	355	84	401
3.	2507	91	58	268
4.	622	477	71	131
5.	84	457	65	35
6.	764	479	48	121
7.	107	452	52	38
8.	30	308	54	19

Figure 6-17. Clustering algorithm example.

In practice, the goal is often to select as many features as possible from an image and to select them in such a way that they maximize the ability of the computer to distinguish between objects. A major consideration is often the speed of process time, which in military flight hardware can be more important than 100% accuracy of identification if the system is supplemented by human supervision. The opposite is generally the case in a factory environment where the system has more time to process, but the lack of human supervision (which is why the machine vision system was specified) demands a higher level of identification accuracy.

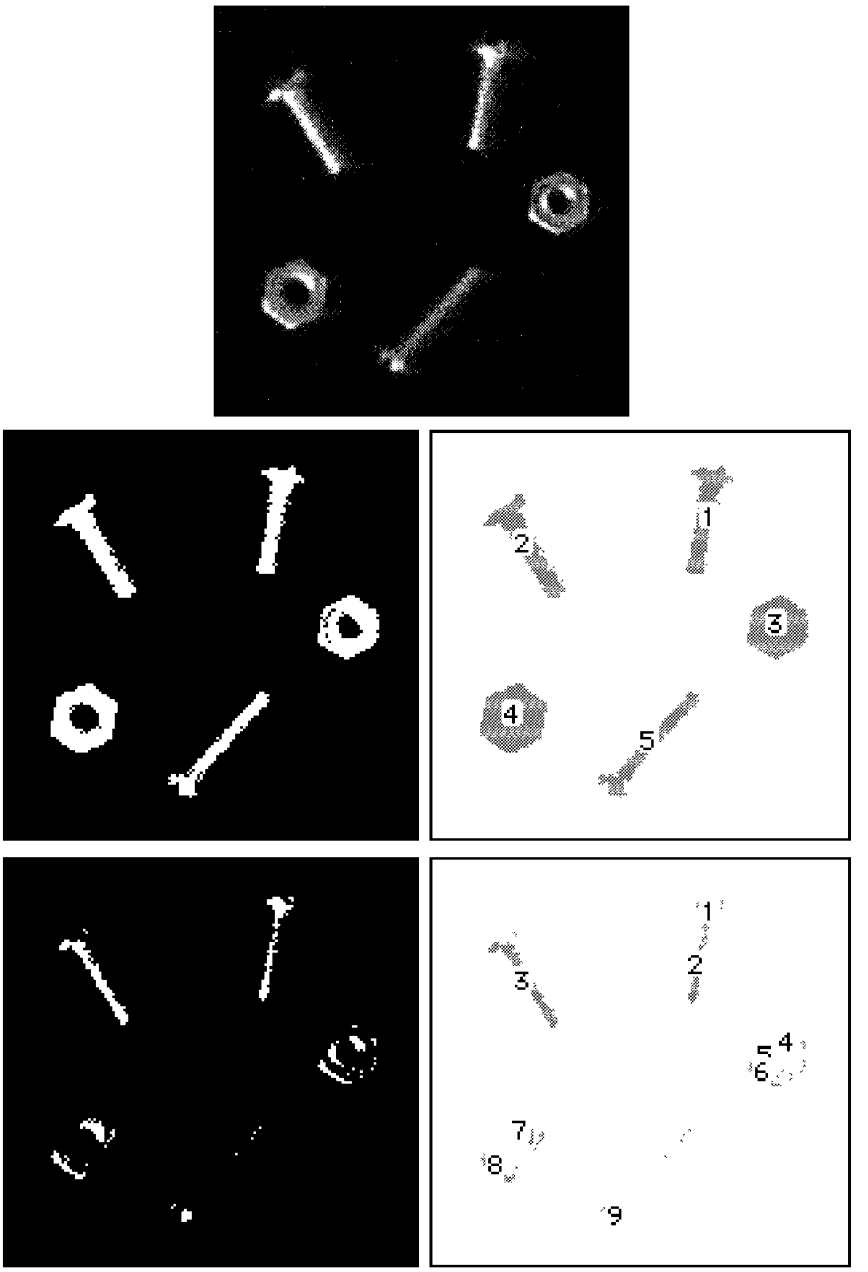


Figure 6-18. Cluster variations from threshold. Top: original image; Middle and bottom left: two different thresholds; Middle and bottom right: results of cluster analysis.

REFERENCES

1. H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," IEEE Trans. Electronic Computers, EC-10, 1961.

2. H. Myler and A. Weeks, *The Pocket Handbook of Image Processing Algorithms in C*, Prentice Hall, Englewood Cliffs, 1993.

3. R. Gonzalez and R. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

4. K. Castleman, *Digital Image Processing*, Prentice Hall, Englewood Cliffs, 1996.

5. E. R. Dougherty, *An Introduction to Morphological Processing*, SPIE Press, 1993.

CHAPTER 7
IMAGE SEQUENCES

In this chapter we examine the processing of image sequences. Image sequence processing is important in robot vision, where the imaging is often done from a mobile platform.

7.1 FRAME-TO-FRAME ANALYSIS

When discussing image sequences, we progress from the realm of image processing that, for the most part, is concerned with still images, and enter the realm of video processing. Video refers to image sequences that are described in terms of frames, the video term for images. A set of frames with an associated rate constitutes a video clip and a set of clips makes up a movie. Movies, clips, and frames first existed on photographic film and later were captured electronically by analog television cameras. Television established the first medium for image sequence processing, so we will begin this section with a discussion of it.

Historically, television signals began as analog scans of image data detected by a vidicon tube. The vidicon processes image data as a continuous signal in time by scanning the image focused onto a target screen with an electron beam. Scanning starts at the upper-left corner of the image and scans across one row (or line) at a time. This scanning is called rasterizing and the analog signal produced—image intensity in time—is decoded with timing signals that indicate end-of-line and end-of-frame. The frame rate is the number of frames per second (fps). For television, the frame rate is roughly 30 frames per second. It should be noted, however, that television consists of two *interlaced* frames. The total number of lines in a television image is 480. Each interlaced frame, consisting of 240 lines, is output at a rate of 63.5 microseconds. The fusion rate of the eyes and the persistence of the television receiver phosphors combine to yield a continuous image sequence in time with no flicker.

Color television cameras employ three vidicons, one each for the red, green, and blue components of visible light. The color television signal, however, is not as simple as three red, green, and blue vidicon signals in tandem. Instead, color television separates the color, or chroma, information from the intensity, or luminance, information. This format was determined by the National Television Standards Committee so that color television signals

would be compatible with monochrome signals, hence the color television signal is called NTSC. The standard for the monochrome signal previously described is RS-170.[1]

RS-170 and NTSC signals can be digitized into individual frames and stored. Formats exist for the generation and storage of digital movies and these will be discussed in Section 7.2, Data Management. All of the previously discussed algorithms for machine vision can be applied to digital movies. However, a number of algorithms exist that take advantage of the fact that movie sequences contain objects that change position from frame to frame and the special class of information that motion conveys. The most popular of the image sequence algorithms, and potentially the most important to machine vision applications, are the tracking algorithms, which are covered in the next section.

There are three methods of digital image sequence acquisition: *flying- spot scanning*, *line scanning,* and *imaging arrays*. The flying-spot scanner has limited and specialized application and consists of a single detector that is mounted to a gimbal mechanism that allows it to scan horizontally and vertically. Line scanning is the mechanical analog to electron-beam scanning where the image data is acquired one line at a time. Like the flying-spot scanner, line scanning has specialized applications, primarily in high-resolution imaging, and will also be covered no further. Unlike the vidicons discussed earlier, imaging arrays capture and digitize an entire image at once. The pixel data is collected from individual sensors in the imaging array and delivered directly to memory. Nevertheless, each of the sensors is polled for its associated image data value and transferred to memory one pixel at a time, much like the analog scanning that takes place in the vidicon. It is possible to combine image memory and imaging sensors as one unit and by doing so the camera avoids the necessity to serially scan for the data transfer from sensor to memory.

Frame rates for digital cameras are determined from the speed of the hardware, which includes both the imaging array and the frame memory. Early high-resolution cameras had fairly slow frame rates due to the bandwidth of the data channels connecting them to the computer and the large amount of pixel data that had to be transferred. As with other applications, the use of image sequences in machine vision problems is highly dependent on the imaging hardware. In order for the vision system to detect features necessary for the application, it is important that the resolution required for spatial sampling and pixel quantization be specified adequately. This is no different from the specifications on still image analysis, but now the processing capabilities of the computer and the amount of data (roughly proportional to the number of pixels) that can pass through the system will impact how many frames can be processed in an image sequence. Depending on the hardware, processing the desired frame rate for the required resolution and pixel depth may or may not be possible.

Whether or not image sequence processing is needed at all will be determined from the overall goal of the system and whether or not it must

track, follow, and/or navigate in real time. The frame rate necessary for these activities will depend on the speed of the object(s) being tracked and the motion of the camera platform. For example, when building autonomous, walking, robots the frame rate is substantially less than that required by the aircraft tracking systems in high performance aircraft. These items

- resolution (sampling, image size)
- quantization (pixel depth)
- frame rate

must be considered when designing or specifying a machine vision system that must analyze image sequences. If the algorithms developed for the system require 50 milliseconds to process an image, then the maximum frame rate possible will be 20 frames per second. This may be acceptable to the walking robot that crawls like a chameleon, but the aircraft system will demand better performance. If the designer opts for lesser resolution and quantization, then the system processing time per frame will decrease and the frame rate will increase. However, less resolution may mean that features, and ultimately objects, will be unresolved. This will decrease system performance. Some applications must be content to just wait for the technology of image processing hardware to improve.

7.2 IMAGING TRACKERS

The tracking of objects is an important component of a machine vision system and is one of the fundamental methods of image sequence processing. There are many different kinds of digital imaging trackers, and we will discuss four of them in this section: differencing, correlation, centroiding, and gated video. Trackers not covered here are the reticle, scanning, and pulsed laser trackers. An excellent treatment of these can be found in Ref. 2.

The goal of imaging trackers is to follow an object's position from frame to frame. It is important for the tracker to first acquire the object. Sometimes this task is performed by a human operator. It is equally important for the tracker to locate the object in the next frame. If the object moves too quickly, the tracker can lose the location of the object. In military terminology, we say that the tracker has lost *lock*. When lock on an object has been lost, the tracker enters a *search* mode as it tries to acquire the object again. If the tracker searches aimlessly, it is said to be *hunting*. Frame rate and the frame processing rate will determine how fast an object can move and still be tracked. For example, the frame rate of a camcorder (30 frames/second) would not permit tracking of a bullet fired from a pistol. Additionally, if you have the frame rate needed to keep up with a given object, but processing of each frame requires so much time that multiple frames have been generated, then the data in those frames will be lost.

7.2.1 DIFFERENCING TRACKERS

The simplest and easiest to implement of the imaging trackers are the differencing trackers. The basic idea is to subtract two sequential frames and note the difference. If the frames are identical, then the difference image will be zero and nothing has moved from the first frame to the next. Of course, something *could have moved* in between the time taken to digitize the first frame and to digitize the second, but that is a problem of selecting adequate hardware to accommodate the speed of objects to be detected.

Figure 7-1 illustrates the basic principle of differencing an image sequence. The first six frames are part of a movie clip of the author blinking slowly. The frames were captured at 15 per second, a resolution of 200 × 120 pixels, with 16 levels of grayscale. The frames are sequenced starting in the upper left and moving from left to right. The five frames at the bottom of the figure, labeled difference sequence, are the result of subtracting each of the frames from the previous frame. Each difference frame shows varying amounts of motion from the previous frame. No motion occurred between the last two frames, so the difference is blank. This type of differencing is used by security firms to monitor warehouses and other facilities as a motion detection method. It is a simple algorithm to evaluate the difference frame to determine if motion has occurred. It consists of looking at every pixel to determine if it is other than white. The algorithm can get more complicated if vibration causes motion errors. This problem can be overcome by applying thresholds to the difference values. Also, if the imaging hardware includes a real-time histogram generator, then the histogram of the difference image can be used to detect motion.

To extend the concept to tracking, we need only compute the centroid (Section 5.3, Mensuration) of the entire difference image. This will yield the X-Y coordinates of the moving object. The advantage of differencing prior to tracking is that the background is eliminated. Of course, this assumes that the camera and background are not moving and that the frame-to-frame correlation of the background is maximum. What this means is that if the camera is noisy, then a difference operation will produce data that represents the noise and the tracking will be disturbed.

7.2.2 CORRELATION TRACKERS

The simplest way to think of correlation trackers, unlike differencing trackers, is that they seek to detect similarity from frame to frame. A correlation tracker computes the object displacement as it moves within the frame by computing the correlation between a reference frame and the current frame. The reference frame must contain the object being tracked.

The 2-D discrete correlation equation is given by:

$$g(x,y)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} frm(m,n)ref(x+m,y+n),$$

where $g(x,y)$ is the result, $frm(x,y)$ is the current frame, $ref(x,y)$ is the reference and M and N are the dimensions of the images. The correlation process is also called template matching, where the reference image is the template to be matched in the current frame.

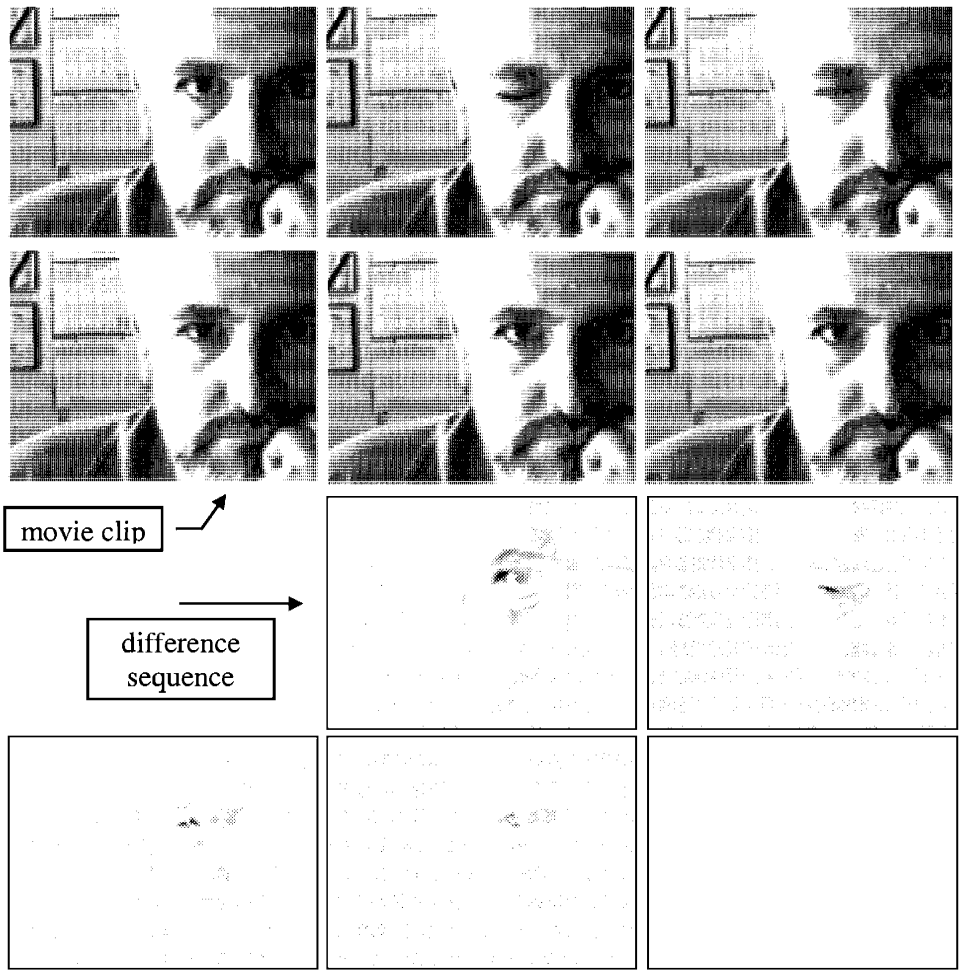
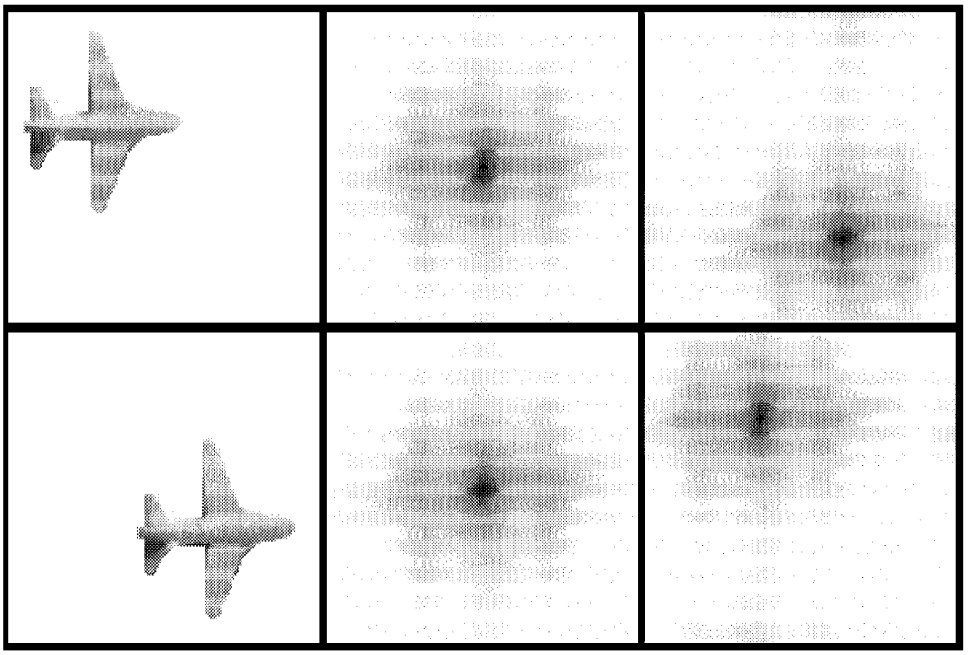


Figure 7-1. Difference frames.

Figure 7-2 shows a set of correlation results. The first column shows two planes displaced from each other. The second column shows the *autocorrelation* of each of the planes, respectively. Autocorrelation means that the image was correlated with itself. This image has important usage in laser



Plane 1	Autocorrelation Plane 1	Correlation Plane 1→Plane 2
Plane 2	Autocorrelation Plane 2	Correlation Plane 2→Plane 1

Figure 7-2. Correlations of two images of an airplane shifted in position.

imaging studies, but here we show it to illustrate that the autocorrelations of the two planes is identical, in spite of the displacements. The final column shows the correlation of Plane 1 with Plane 2, then the correlation of Plane 2 with Plane 1. In both cases, the correlation "finds" the first image within the second; i.e., the upper correlation (last column) finds the location of Plane 1 in the Plane 2 image, and *vice versa* with the second correlation below it. This illustrates well the mechanism of the correlation tracker. By thresholding the correlation we determine the pixel coordinates of the object and thus achieve tracking.

Correlation trackers have two distinct advantages: (1) they track well in noisy environments and (2) they track while field of view is expanding. Both of the advantages are in contrast with the centroiding, or gated trackers, which

are discussed in the section to follow. The disadvantage of the correlation tracker is that it requires an external agent to designate the object to be tracked.

7.2.3 CENTROID TRACKERS

The centroid of objects computed from frame to frame can be tracked to determine the positional behavior of an object. The centroid can be computed following a cluster analysis (Section 6.2) and multiple objects within a set of frames can be tracked. Trackers that operate in this fashion do not require designation by an operator. The tracking will be optimal, however, whether or not the object being tracked is desirable to the user will depend on how robust the segmentation and recognition processes are.

The centroid algorithm has been shown[2] to be optimal with respect to tracking capability using maximum likelihood arguments. The reader is left to seek the proof in the reference, however, it is easy to see that an empirical argument can be employed to demonstrate that the centroid algorithm is optimal in finding the center of an object. Consider a flat plate of some stiff material of uniform thickness and density. Balance the plate on a fulcrum. The balance point of the plate will be at the center. Now apply a small weight to some point other than the center. To maintain balance you must shift the fulcrum towards the weight. Now think of the weighted plate as an image, where the weight (or weights) is proportional to grayscale levels. The centroid algorithm finds the fulcrum point of the weighted image.

To compute the centroid, the first and second moments of the object must be determined. This is done with the following formula:

$$m_{ij} = \sum_{x} \sum_{y} x^i y^j f(x, y)$$

and the centroid coordinates are then:

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}}$$

Figure 7-3 shows an object against a grid. If we compute the moments on the object (assuming a value of 1 for a black pixel and zero for white) the value of m_{00} is 33 (there are 33 black pixels in the object), m_{10} is 210 (sum all the x coordinates of black pixels) and m_{01} is 121 (sum all the y coordinates of black pixels). The centroid coordinates are then (6,4). This may appear inconsistent, but the result is normalized to the object's origin which is at coordinates (20,11).

The algorithm to determine the centroid of an object is fast and simple, yet it does not always reveal the geographic center of an object, which may be preferred for targeting applications. In this case, the computation of maximum and minimum axes will locate the geographic center. This value can also be

tracked, however, the computations are substantially more complex than those for the centroid.

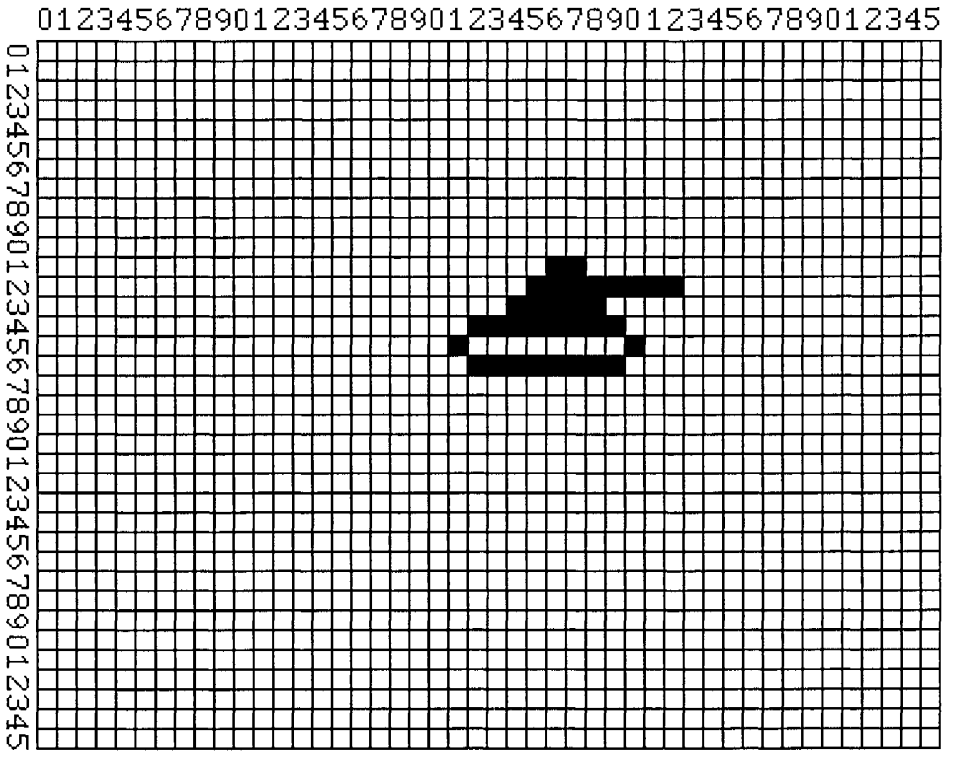


Figure 7-3. Centroid.

7.2.4 GATED VIDEO TRACKERS

Gated video tracker algorithms were first implemented in analog hardware, hence the name. They were developed to process peaks detected by scanners, where the "gate" was a threshold level indicating the presence or absence of the target. Although computer algorithms for the gated video tracker exist in many variations, they all have two primary components in common, the image frame and the gate. The gate is also known as the tracking window. An illustration of these parts is shown in Figure 7-4.

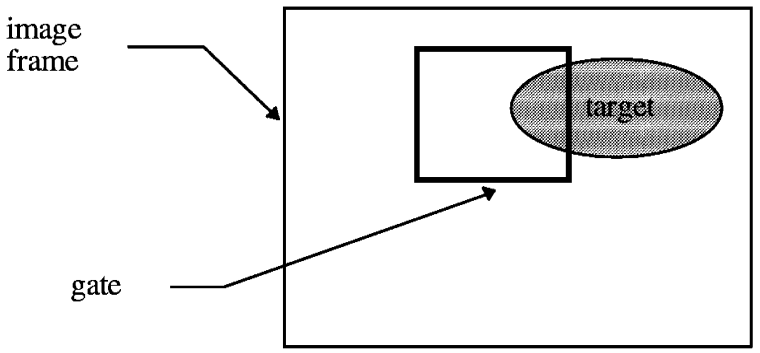


Figure 7-4. Gated Video Tracker components.

The gate changes in size to accommodate the size variations of the target. Geometry of the gate can vary and algorithms controlling the gate try to exploit various features of the target or the hardware, or to reject noise. The algorithm starts either by *searching* for a target or by being positioned over the target by a human operator, or by receiving target coordinates from another source. A system can have different search modes—the simplest puts the gate in a corner of the image and causes it to scan back and forth until a target is located. Another option is to have the gate start at the center of view and move in a circular pattern outward towards the edge of the image frame. A third pattern is one in which the gate samples image pixels randomly.

The tracker tracks by comparing statistics of the target (or the object that it perceives to be the target) with those of the image frame, or background. The goal of the algorithm is to maximize the number of pixels within the gate with target statistics. These statistics can be as simple as binary values or as complex as multiple features. Figure 7-5 shows three binary images of an object (not moving) and the corresponding value of target pixels in the gate computed by a gated video tracker. The gate depicted in the center frame is on target. As the target moves under the gate in the next frame, the pixels in the vicinity of the gate are analyzed and the gate is moved to keep the maximum number of target pixels within it. Although we imply a gate that is fixed in size, it can be made to change in size to keep the target centered. The location of the gate in the image frame indicates the geographic center of the target.

A more advanced configuration used in gated video tracking is shown in Figure 7-6. Here the gate has been partitioned into a target window, a background window, and corner windows. Statistics are computed for pixels within the different windows and compared against prior frame statistics. These statistics include background pixels (pixels that are completely outside of the windows and contain no target pixels) and target pixels that are both in and out of the target window. In this system, the gate has been replaced by the windows. The window is resized as these statistics change while the algorithm seeks to keep pixels with target statistics in the target window and pixels with

background statistics in the background windows. The corner windows are used to rapidly detect direction changes of the target while the background window's function is to size the target within the target window. The location of this window in the image frame yields track location data.

The gated video tracker is easy to implement in real time and is used in high-performance aircraft targeting systems. The principal drawback of the tracker is that it is highly susceptible to noise and distraction (track can easily shift to an object passing in front of the tracked object).

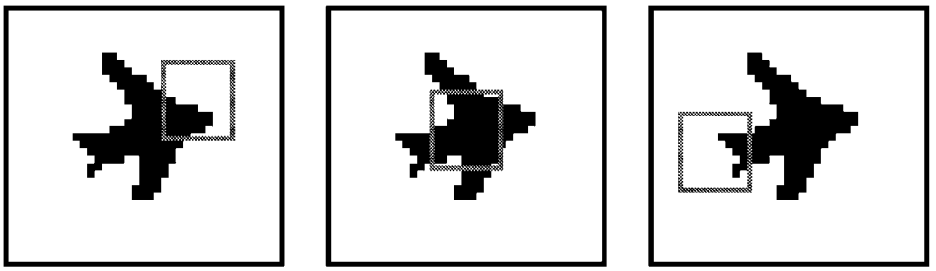


Figure 7-5. Gated Video Tracker gate values. (numbers under each frame indicate number of target pixels in each gate).

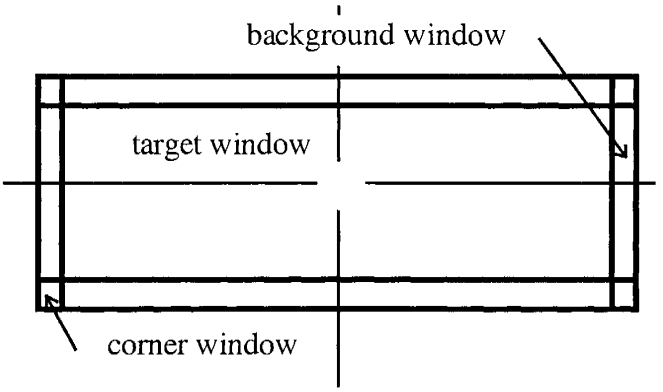


Figure 7-6. Gated video tracker with target/background window geometry.

7.3 DATA MANAGEMENT

The volume of data to be managed when working with image sequences is large, particularly during development efforts. Once a system has been developed, the need for storing image sequences is generally over. Image sequences are stored as movies and because of the huge amount of data, a number of storage schemes with compression have been implemented. The most popular of these are MPEG, QuickTime™ and VFW™.

MPEG is an acronym for the Motion Pictures Experts Group. This is an independent standards committee that has established the MPEG video format which defines a bit-stream representation for synchronized digital video and audio, compressed to fit into a bandwidth of 1.5 Mbit/sec. The following are MPEG features:

- Corresponds to the data retrieval speed from CD ROM and DAT, and a major application of MPEG is the storage of audio visual information on this media.

- The MPEG standard is the three parts—video encoding, audio encoding, and "systems" which include information about the synchronization of the audio and video streams. The video stream takes about 1.15 Mbit/sec, and the remaining bandwidth is used by the audio and system data streams.

- MPEG video encoding starts with a fairly low-resolution (352 × 240 pixels × 30 frames/s) video picture. RGB pixel information is converted to chrominance/luminance and a complex, lossy compression algorithm is applied. The algorithm takes the time axis as well as spatial axes into account, so a good compression ration is achieved when the picture is relatively unchanging (and vice versa).

- Compressed data contains three types of frames: I (intra) frames are coded as still images; P (predicted) frames are deltas from the most recent past I or P frame; and B (bi-directional) frames are interpolations between I and P frames. I frames are sent once every 10 or 12 frames. Reconstructing a B frame for display requires the I and/or P frames, so these are sent out of time-order.

- Substantial computing power is required to encode MPEG data in real time—perhaps several hundred MIPS to encode 25 frames/second. Decoding is not quite so demanding.

- The quality of MPEG-encoded video has been compared to that of a VHS video recording.

QuickTime™ is Apple Computer's video format. QuickTime™ version 2.0 supports time-code, 60 fields per second video (640 × 480) and high-data throughput greater than 3 MB per second. This represents a 300% increase over previous versions of QuickTime. The implementation of QuickTime™ on the Macintosh platform makes it much easier for computer users to create, edit, play back, and synchronize music with video—all without a technical understanding of the "Musical Instrument Digital Interface," or MIDI, technology. The ability to store the musical scores for instruments saves disk space for users, because QuickTime™ music tracks are much smaller than digital audio. For example, Beethoven's 5th Symphony requires 300 MB if stored as CD-quality audio, but when represented as a QuickTime™ music track, it needs just 800 K. The compression-decompression (CODEC) scheme used by QuickTime™, as with Video For Windows™ (discussed below), is user selectable.

Audio Video Interleaved (AVI) files constitute Microsoft's Video for Windows (VFW™) video data format. Microsoft Video 1, Microsoft RLE, Microsoft Full Frame, and Cinepak by SuperMatch CODECs are supported. Video for Windows creates CD-ROM movie files that play back at 150 KB per second, the standard CD-ROM data transfer rate for multimedia personal computers. Video for Windows normally interleaves audio with every frame of video.

MPEG, QuickTime™ and VFW™ are video storage standards that are continually being improved. Although MPEG video is compressed to a specific CODEC, the QuickTime™ and VFW™ formats use CODECs that are user selectable. A listing of various CODECs for Microsoft and Apple platforms is given in Figure 7-6. There are a number of approaches that can be taken to the image compression. These are:

- wavelet—basically a windowed transform (Hartley) method.
- DCT—Discrete Cosine Transform.
- VQ—video quantization, a subsampling method.
- Fractal—uses geometric primitives derived from Mandelbrot sets.
- RLE—Run Length Encoding.

Using the video storage schemes outlined here requires specialized software to extract the image data one frame at a time. Software is available that allows processing of individual image frames in movies. One major consideration when using video storage methods for machine vision work is whether or not the compression method is *lossy*. This means that when the image is decompressed, some data from the original frame is lost as it is judged redundant by the compression algorithm. This redundant data may not be

redundant to the vision algorithm and indeed may contain critical information. Therefore, caution must be taken when using lossy CODECs. Most software offers the user a choice of compression and software that has variable lossiness, such as Cinepak, can be set for no loss at all. Of course, the compression efficiency will suffer. QuickTime and AVI formats allow storage of image sequences as individual JPEG compressed files. JPEG permits lossless compression at a substantial penalty over the lossy mode.

The computational burden of compression algorithms has given rise to specialized hardware that compresses and decompresses in real-time (30 fps or better) rates. CODECs that are hardware-based are indicated in Figure 7-7 as those without a bullet in the "software play" column. Note that none of the CODECs that exceed 30 fps are capable of software playback. Also note that the maximum size (pixels) of the images can be restrictive. All of the CODECs and listed data are based on color images.

CODEC	Company	Approach	Platform	Soft-ware Play	Size	Rate
Captain Crunch	Media Vision	Wavelet	PC		320x240	30
Cinepak	SuperMac	VQ	Mac,PC	•	320x240	15
DVI-RTV	Intel	VQ	PC		256x240	15
DVI-PLV	Intel	VQ	PC		640x480	30
Indeo	Intel	VQ	Mac,PC	•	320x240	15
Motion-JPEG	n/a	DCT	Mac,PC		640x480	60
MotiVE	Media Vision	VQ	PC	•	160x120	12
MPEG 1	n/a	DCT	PC		320x240	15
MPEG 2	n/a	DCT	PC		704x480	60
Px64	n/a	DCT	Mac,PC		352x288	15
Pro-Frac	TMM	Fractal	DOS	•	320x200	30
SoftVideo	TMM	RLE	PC	•	640x480	15
Ultimation	IBM	n/a	OS/2	•	320x240	30
Video	Apple	VQ	Mac	•	160x120	15
VideoCube	ImMIX/Aware	Wavelet	Proprietary		640x480	60

Figure 7-7. Common CODEC table.

If consideration for the issues mentioned above is taken into account, machine vision systems can find video compression useful in applications where the loss of image data is not critical. A new research area has been developed that seeks algorithms that can process images in their compressed state. Computing in compression algorithms shows promise in that the compressed image sequence has, by definition, had unnecessary or redundant data removed. If an algorithm can process the reduced data, then a substantial improvement in process time can be achieved assuming that the image was compressed in real time (a feat easily done using hardware compression units).

REFERENCES

1. L. Galbiati, *Machine Vision and Digital Image Processing Fundamentals*, Prentice Hall, Englewood Cliffs, 1990.

2. G. Gerson and A. Rue, "Tracking Systems," in *The Infrared Handbook*, W. Wolfe and G. Zeiss, eds., 3rd Printing, Office of Naval Research, U. S. Navy, Washington, 1989.

3. H. Myler and A. Weeks, *The Pocket Handbook of Image Processing Algorithms in C*, Prentice Hall, Englewood Cliffs, 1993.

CHAPTER 8
VISION SYSTEMS

It is appropriate that this final chapter is dedicated to exploring general- and special-purpose vision systems. These systems attempt to incorporate all that is known about machine vision (at the time of their development) and they have all been successful in some way. Certainly they have added to the collective knowledge of vision systems in general. The first section of this chapter is a general survey of systems that have been constructed to do complex machine vision tasks followed by a section on model-based vision systems, followed by a discussion of two advanced systems that incorporate recognition by components theory. The chapter concludes with a discussion on the development of vision systems.

8.1 SURVEY

The survey of vision systems dates back to the mid-1970s when image processing algorithms were well established and computer hardware had become sophisticated enough to handle complex algorithm suites. The survey is presented as a chronological list with the important features or discoveries of the systems tabulated. The number of systems that has been constructed since the 70's is legion and this survey is more historical than taxonomically complete. The literature abounds with complex and detailed descriptions of the attempts of research groups to solve the general machine vision problem. This survey will provide the reader with a chronology of complexity and an idea of the depth of knowledge required to construct a general purpose system. Appendix A provides a source of software and resources on the World Wide Web where many parts of the listed systems may be found in one form or another.

Garvey and Tenenbaum[1] (1974)

- goal was to locate objects in an office environment where all objects were known to the system
- data input:
  - range to object
  - reflectivity at one wavelength

- RGB images
- strategy was to:
  - acquire image samples that might belong to the object
  - validate the hypothesis for which object was being viewed
- used simple local features and contextual relations

Bolles[2] (1977)

Verification Vision (VV)

- used image and object models
- was intended for inspection and visual control in repetitive manufacturing tasks
- made use of 3-D models
- objects could not have major shifts in appearance or be occluded by other objects

Shirai[3] (1978)

- heavily dependent on edges
- straight lines and ellipses were used to describe edges
- edges classified as line, circle, or ellipse
- small gaps filled (defragmentation)
- analysis began with most obvious object, next most obvious, etc.
- found most obvious feature, etc.
- e.g., for lamp-> finds lamp shade -> looks for trunk-> looks for base

Ballard[4] (1978)

- used image models to locate ribs in chest x-rays
- system structured in three levels:
  - the model
  - the sketch map synthesized during image analysis that related the model and the image
  - image data structures
- similar to VISIONS (see below), except segmentation level was established by query
- was not intended as a general vision system

Nagao and Matsuyama[5] (1980)

- goal was processing of aerial photos
  - edge-preserving smoothing (defragmentation)
  - regions of continuous spectral properties located
  - extracted "cue" regions (clusters):
    - large homogeneous regions
    - elongated regions
    - shadow/shadow making regions
    - water regions
    - high-contrast texture regions
    - shadow detection
  - brightness histogram processed
  - used bimodal threshold
  - homogeneous regions that were less than the threshold and were classified as shadows
  - shadow-making regions adjacent to shadows and away from sun
  - elongated objects were detected from skeletons
  - houses/manmade structures—recognition was based on rectangularity, roof detection, location cues

The common element among each of the systems listed is that they perform the sequence of thresholding, segmentation, representation, and classification as described in this book.

8.2 KNOWLEDGE-BASED VISION: VISIONS, ACRONYM, AND SCERPO

Knowledge-based approaches to machine vision, particularly expert systems, became popular in the late 70's and 80's during the heyday of artificial intelligence. Knowledge-based systems are computer programs that access a database of knowledge. Knowledge in this context can be loosely defined as facts about facts. For example, a system may have concluded that an object is a small lamp but it is not clear as to what the object is that the lamp is sitting on. A knowledge base may have data that indicates that small lamps (generally) sit on tables. A knowledge-based system can use this knowledge to conclude that the object the lamp is sitting on is a table. Of course, the dilemma is the *generally* modifier in the statement "small lamps sit on tables."

The difficulty that knowledge-based systems face is the size of their domains. If we include all of the possible images that can be observed, the size of the domain becomes astronomical. Consider the following thought experiment using Figure 8-1. The image shown in the figure is 320 × 240

pixels at 16 levels of grayscale. Even after laser printer dithering, the image clearly shows detail and complexity. It is not difficult, however, for the adult human to see that there are three simple wooden shapes in the foreground, sitting on a cluttered desk. One can make some simple deductions about the scene such as the fact that the objects are illuminated from the left, there are pictures on the wall mounted in glass frames, etc. The "knowledge" here is that the swirled patterns we detect on the shapes lead us to conclude that they are wooden, the bright reflections on the pictures tells us that the frames have glass covers, and the piles of papers reveal the cluttered aspect of the desk. These deductions may seem simple to us, but to include them in a vision system is no simple task.

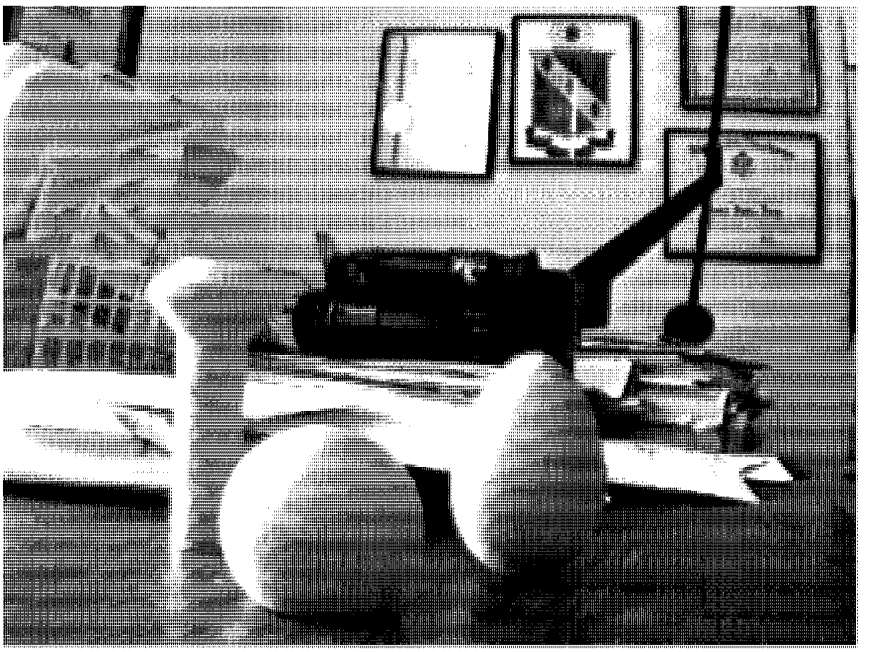


Figure 8-1. Objects on desk.

The extraction of objects is understandably difficult, but when you consider the range of possible knowledge facts that pertain to office environments, then extend that thought to all possible image scene contents, the size and complexity of the knowledge base required becomes staggering. Three systems, VISIONS, ACRONYM and SCERPO, will be examined by virtue of their popularity, complexity, and their hubris in attempting to interpret *any* natural scene. SCERPO is also *model-based* in that it attempts to identify objects by matching their component parts to models of objects stored in a database. It is included here (instead of the next section) because it is

typically listed as a knowledge-based approach and does not use a component-based modeling scheme.

Hanson and Riseman[6] (1978) VISIONS

(Visual Integration by Semantic Interpretation of Natural Scenes)

- multiple levels of interpretation of information on particular images and data stored in a long-term knowledge base.
- modular knowledge sources that transform information from one level to another.
- hierarchical structure.
- bottom-up and top-down reasoning paths.
- operates on outdoor scenes and maintains partial models of previously-observed objects.

Brooks[7] (1981) ACRONYM

(ACRONYM is not an acronym, just a play on systems with acronyms!)

- reasons from first principles
- based on algebra and projective geometry
- uses viewpoint-independent 3-D object models in the form of symbolic expressions with numeric type.
- searches for instances of models in images.
- predicts appearances of models in terms of ribbons and ellipses.
- high resolution imagery.

Lowe[8] (1987) SCERPO

(Spatial Correspondence, Evidential Reasoning and Perceptual Organization)

- model-based recognition.
- 3-D shapes of objects are directly obtained from features of their 2-D shapes.
- Objects described by polyhedral parts.
- Recognition achieved by matching.

Knowledge-based systems have given way to model-based approaches. Knowledge-based systems operate on the assumption that knowledge (facts about facts) can be encoded in a database and applied to situations in order to reason about what is happening. Model-based methods are linked to perceptual

studies of humans. These studies have suggested that humans use 3-D component groupings to identify complex objects. Model-based vision is in an infant stage because the systems that employ it seek to identify objects rather than scenes.

8.3 MODEL-BASED VISION: VITREO AND PARVO

Model-based vision systems process image data and then try to fit a model of known objects to what has been extracted from the unknown scene. There are various ways to do this—possibly the most popular are systems that extract Computer Aided Design (CAD) data from images in the form of perspective views—SCERPO being the most well known. The extracted data is then rotated, distorted, or resized to try and match objects stored in a database. Model-based systems can also seek to relate other forms of data such as features of known objects. We discuss two very complex and powerful model-based systems that are predicated on recognition by components theory.

Recognition by components (RBC) theory was first proposed by Biederman[9] as a means of modeling the human visual system. RBC models vision as a 3-D system that decomposes objects into unique components that Biederman called *geons*, for geometric objects. Geons are constructed from surfaces of revolution and have specific properties. There are 36 geons (see Figure 8-2 for two examples), and from these Biederman has estimated that the number of objects possible would exceed two million. Two systems have been constructed using RBC theory, VITREO and PARVO.

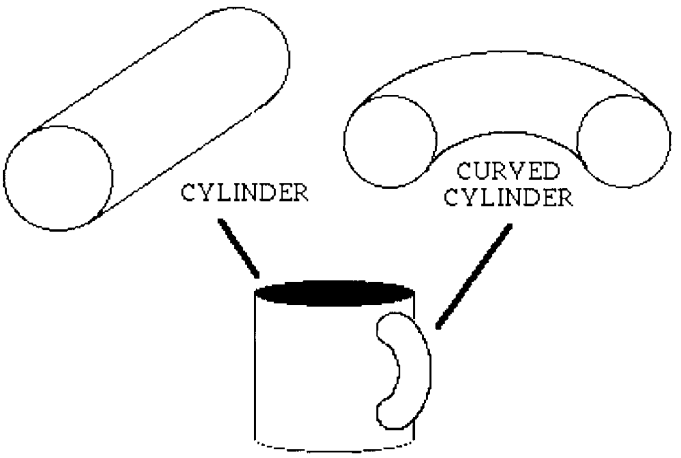


Figure 8-2. Geons with object created from them.

VITREO[10] or Viewpoint Independent 3-D Recognition and Extraction of Objects is a model-based general purpose vision system that is based on RBC theory. VITREO incorporates multiple processing stages via a bottom-up analysis meaning that it processes the object image for lines and then attempts

unique object recognition by non-accidental view and restoration of occluded parts. A major feature of VITREO is that it uses grayscale images for input.

PARVO[11] or Primal Access Recognition of Visual Objects, preceded VITREO and also uses RBC theory. Like VITREO, PARVO incorporates multiple processing stages but proceeds from a top-down analysis and uses line drawings of objects for input. Both PARVO and VITREO can be classified as model-based systems because they access a database of object models after decomposition of unknown objects into components. The model-based approach is an outgrowth of knowledge-based methods and is believed to be the fundamental mechanism of human perception of objects.

8.4 BUILDING A MACHINE VISION SYSTEM

We now list a set of vision system objectives:

High-performance:

- complex scenes with many objects, high detail, accuracy, resolution, and speed.

Generality:

- generic with respect to object class and observation.

Completeness:

- should span all application tasks.
- implies requirement for powerful perceptual mechanisms.

Intelligence:

- reasoning in the domains of images and surfaces.
- similar to a human observer or analyst.

These objectives describe, in essence, the capabilities of the human visual system. To try to emulate and reproduce them is the ultimate goal of machine vision. Of course, from our earlier summaries we have found that this has not been accomplished. Although it may not have been clear from those summaries, no machine vision system yet developed incorporates all of the objectives listed above. Our final discussion will focus on the answer to two questions. First, "Is there a clear explanation as to why machine vision systems have not reached human-level capabilities?" and second, "What is needed to achieve this level?"

To determine why a machine vision system of human-level capability has not been devised we first have to examine the nature of the problem. When computer systems are designed to tackle engineering problems, one starts with

a mathematical model. The closer this model is to the underlying mechanisms of the problem, the more robust the computer system solving it will be. For example, computer programs for the design of computer circuits are so robust that no modern computer can be designed without them because of the level of complexity of the circuitry. When we look at what is known about the workings of the human mind, which, from Section 1.1 we know is directly coupled to our visual system, the information is sadly lacking. Exact mechanisms of processes are simply not known. The RBC theory proposed by Biederman, although attractive on the surface for componential recognition, is based on observation of psychoperceptual *behaviors*, not on specific neural circuitry studies. There are ongoing debates within the artificial intelligence research community regarding mechanisms and no clear theory has emerged. Right now no model exists that exactly reproduces the human visual system.

From the previous paragraph the reader may surmise that the answer to the second question is that we need a better model. A more appropriate answer to the individual who needs sophisticated vision now is that the problem needs clarity of definition. By this we mean that until a more robust model of human vision has been determined, we must satisfy ourselves with analyzing pieces of the problem. The fundamental problem that co-exists with the lack of a robust model is the fact that basic Von-Neumann architectures simply cannot process the vast amount of data that must be addressed within a complex visual scene. If the domain of the vision system is sufficiently restricted, then it is entirely possible to develop machine vision systems that in some cases exceed the capabilities of human vision.

We are far from developing a completely generic vision system that will emulate the human visual system in capability. For one thing, the computer hardware will have to be substantially faster and capable of more data bandwidth than it is now. Nevertheless, it is possible to develop robust and powerful vision systems for specific tasks by judicious modeling of the problem, restricting the domains, and choosing algorithms carefully.

REFERENCES

1. Garvey, T. and J. Tenenbaum, "On the automatic generation of programs for locating objects in office scenes," *Proc. 2nd Intl. Joint Conf. on Pattern Recognition* , Copenhagen, pp. 162-168, 1974.

2. Bolles, R., "Verification vision for programmable assembly," *Proc. 5th IJCAI*, Cambridge, pp. 569-575, 1977.

3. Shirai, Y., "Recent advances in 3-D scene analysis," *Proc. 4th Intl. Joint Conf. on Pattern Recognition*, pp. 86-94, 1978.

4. Ballard, D., "Model-directed detection of ribs in chest radiographs," *Proc. 4th IJCPR*, Kyoto, 1978.

5. Nagao, M. and Matsuyama, T., *A Structural Analysis of Complex Aerial Photographs*, Plenum Press, New York, 1980.

6. Hanson, A. and E. Riseman, "VISIONS: A Computer System for Interpreting Scenes," in *Computer Vision Systems*, A. Hanson and E. Riseman, eds., Academic Press, New York, 1978.

7. Brooks, R., "Symbolic reasoning among 3-D models and 2-D images," *AI*, pp. 285-348, 1981.

8. Lowe, D., "Three-dimensional object recognition from single two-dimensional images," AI, pp.335-395, 1987.

9. Biederman, I., "Human image understanding," in *Human and Machine Vision II*, A. Rosenfeld, ed., Academic Press, Boston, 1986.

10. Yoo, H., *VITREO: Viewpoint Independent 3-D Recognition and Extraction of Objects*, Ph.D. Dissertation, University of Central Florida, Orlando, 1993.

11. Bergevin, R., and M. Levine, "Part Decomposition of Objects from Single View Line Drawings," *CVGIP: Image Understanding*, pp. 73-83, January 1992.

APPENDIX A
SOFTWARE

There is a large amount of image processing software available. Some of it is appropriate for machine vision tasks and some isn't. This appendix is intended for those who are evaluating imaging software for use in their lab or facility. The most prudent strategy is to become knowledgeable about algorithms and imaging using low-cost or free software before investing in a commercial package. There are three major software "platforms" or operating systems. Almost all others are too specialized for consideration here. These platforms are Unix, MSDOS/Windows and Macintosh (MacOS). The software that is available free of charge and that performs a large range of image processing functions will now be discussed. This discussion was culled primarily from an article by the author entitled "How to choose image processing software for your application," which appeared in the September 1994 issue of *Laser Focus World.* Although the article was aimed at users interested in image processing rather than machine vision, it is also useful for machine vision requirements.

Image processing may have first begun with the coding and transmission of newspaper images across transatlantic cable in the early 1900s. The field has grown in complexity and breadth since then with a dizzying array of hardware and software options available to the engineer or scientist in need of imaging support for an application or research effort. Image processing implies *digital* image processing, or the processing of images by a computer. Most everyone knows what a digital image is, but relatively few understand the complexities that surround its acquisition, verification, validation, manipulation, and storage. We want to reduce this complexity by defining image processing in general and by helping you define your needs. At the end you should be familiar with sophisticated image processing software for very little financial outlay.

Acquisition of images can be as simple as downloading files from the Internet or as complex as high-speed imagers attached to expensive digitization equipment. If you simply need images to experiment with, all the public domain and commercial image processing systems come with sample images as part of the package. Image sets are also available on CD-ROM or from national dial-up services such as America Online, Compuserve and Prodigy.

The verification and validation of image applies to the source of the data and what the conditions of acquisition were. Manipulation is where most of the

complexity arises because here we are primarily within the software domain, with few exceptions. Because of the large variety of algorithms available, this aspect of imaging can be quite difficult to navigate. Storage issues are, in actuality, very straightforward unless you are involved in compression algorithm research. The decision of whether to compress or not to compress is fundamentally a cost issue in terms of time, quality, and dollars. The issue of primary concern is the software for manipulation. This software can be intimidating but it is my personal contention that anyone with basic skills in math and engineering can come to grips with image processing techniques. The real danger lies in making the wrong choices for what can rapidly escalate into a major investment of your time and money.

At some point you have to determine what your application or application area is. This may be the least of your problems because this deals with what you are trying to do in an area you are most familiar with—your own. The real issue is whether image processing is needed at all. Technology, as we well know, exhibits a bandwagon effect on occasion and image processing is driven by technology. As computers become faster and less expensive, and display and storage capabilities follow with increasing resolution and size, the availability and feasibility of imaging for a broader range of applications occurs. Here is where your personal sense of caution versus risk-taking comes into play. You have to determine what advantages image processing may bring to your work, if any.

Image processing is a catchall term that actually comes from the more encompassing field of *computer imaging*. The four fundamental areas of imaging are image processing, computer graphics, machine vision, and multimedia. There are fine lines of distinction between these, and these lines are becoming blurred as the areas mature and the technology improves. To see where your work lies, let's look at the basic definitions of these fields:

> *Image Processing:* traditionally works with real scenes and two-dimensional signal data. The image is treated as a 2-D sampled signal and algorithms are applied to enhance, restore, code, or understand the data. Enhancement refers to the improvement of an image for human use whereas restoration is the removal of image degradation. Coding is the reduction of the data by information-theoretic means and is the area where image compression is studied. Image understanding (see Computer Vision, below) seeks to process the real scene into a descriptive data structure.
>
> *Computer Graphics:* basically the inverse of image processing and computer vision where one attempts to computer-create a real scene from a data structure or description. A side aspect of computer graphics is computer art, which seeks to reveal artistic expression using computer graphics as a medium.

Machine Vision: fundamentally, machine vision *is* image understanding. The area has become so complex and distinct from image processing that it has splintered off into an area of its own. Most image processing texts end with a small section on computer vision (read image understanding) while computer vision texts start with a brief section on image processing.

Multimedia: a hot buzzword of the 90's, Multimedia is easily defined as the use of imaging, sound and text (or other media) concurrently as a coordinated entity. Most modern music videos are good examples of multimedia techniques. The popular *morphing* process, which maps the spatial distribution of one image into another over time, incorporates the well-known image processing technique of warping or coordinate translation which was claimed by computer graphics some time ago.

Independent of which area of imaging your application falls into, the underlying thread of all of these areas is image processing. The impact of software on image processing cannot be understated. The selection of software will be influenced directly by the kind of image data that you are processing.

If you are working with multisensor fusion of multiple wavelength spatial sensor data, your attention will not be on multimedia or computer graphics, but on signal-based image processing that leads to computer vision. Specialized data requires specialized equipment and software, but you can still develop expertise and experiment with one of the three public domain packages we will cover later. Each of them reads raw data. This means the image data stored in a file directly from the acquisition system can be read in and manipulated. Since the systems store data in a variety of formats, you can perform file conversion on your data and make it more easily transported by saving it in a well-known format, like TIFF or GIF.

A major assumption of this discussion is that you probably don't have much of a budget to get familiar with computer image processing or it may be that you are not yet ready to commit a major segment of your budget to an unfamiliar area. Just look around the lab for a handy PC, Mac, or UNIX workstation and pull down one of the software packages we describe below off the Net. Once you are past the learning curve using the public domain imaging software, you will probably need to give some hard consideration to the purchase of a specialized system, at least in terms of software. The public domain software can only go so far (you get what you pay for) and ultimately a major imaging project will require investment into a commercially distributed *and supported* imaging software system. After negotiating the morass of image processing terms, methods, approaches, and theories using the free software, you'll be ready to tackle the marketing side of image processing head on. At least when the salesperson says, "This baby can handle a 3×3 unsharp mask in 2 milliseconds" you may know whether or not that capability is necessary to your application.

Three image processing software packages stand out above the rest because of one major factor, price. They are available free and offer an excellent way to prepare to make hard decisions on image processing needs. The packages run on three different hardware platforms and can be accessed using anonymous FTP on the Internet and from other sources.

For the PC, the UCFImage© package written by Harley Myler and Art Weeks at the University of Central Florida runs under MS-DOS and requires a color VGA monitor. The package was developed and released as "imaging for the masses" because it will run (fast) on computer hardware that costs less than $1,000. The program is included in the back of the introductory imaging text, *Computer Imaging Recipes in C*, published by Prentice-Hall. UCFImage© allows spatial, frequency, nonlinear and adaptive filtering, warping, split-screen, object recognition, morphological operations, pseudocoloring, noise generation, and histogram operations (see Figure 1). Although substantially more modest in capability than Khoros, which is discussed below, UCFImage© allows very sophisticated image processing anywhere you have a PC that supports GIF, TIFF, BMP, and PCX image file formats.

Figure 1. UCFImage© main screen with image display.

For the Apple Macintosh™, NIH Image, written by Wayne Rasband at the National Institutes of Health, was once compared favorably in a *MacWorld* review, to commercial imaging software costing hundreds of dollars. NIH Image uses the friendly and intuitive Macintosh graphical user interface (see Figure 2) and allows a wide range of image processing operations from spatial filtering to multiple-feature object analysis and mensuration. The program is geared towards medical image analysis but anyone doing basic image

processing research and development on a Macintosh will find it invaluable. NIH Image supports PICT, MCID, and TIFF image file formats.

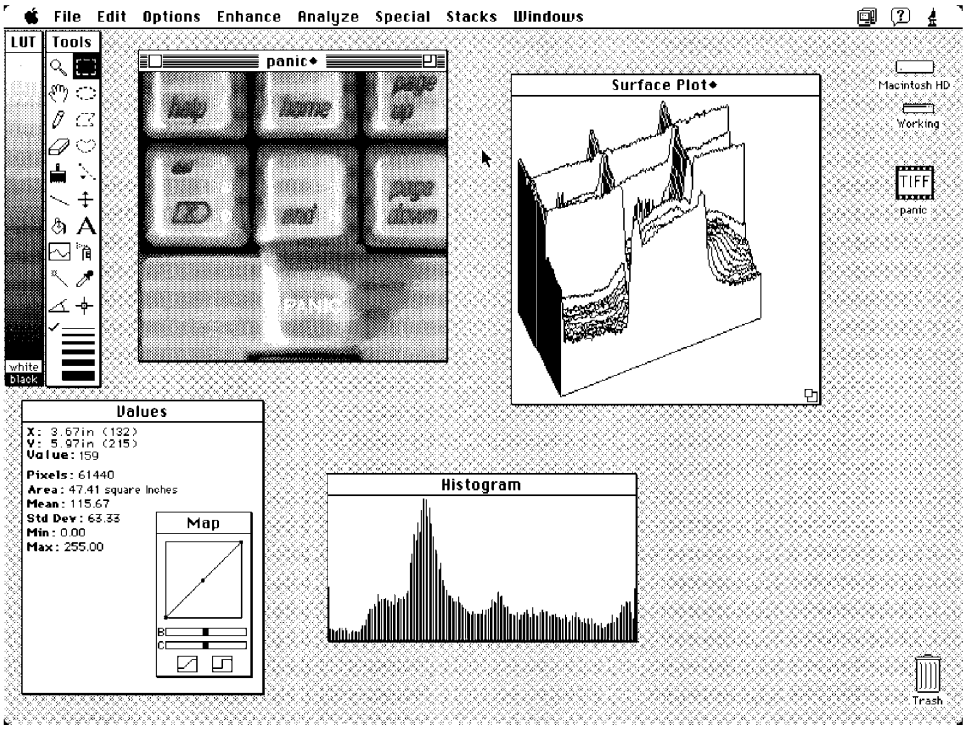


Figure 2. NIH Image main screen.

For X Windows systems, the Khoros package originally developed at the University of New Mexico is an outstanding package. The Khoros system consists of individual routines that run under the UNIX operating system. There are over 260 programs that perform arithmetic, classification, color conversion, data conversion, file format conversion, feature extraction, frequency filtering, spatial filtering, morphological filtering, geometric manipulation, histogram manipulation, statistics, signal generation, linear operations, segmentation, spectral estimation, and transforms. Khoros supports TIFF, pbm, BIG, DEM, DLG, ELAS, FITS, MATLAB, Sun raster, TGA, and xbm image and data file formats.

Most labs have at least one of the computers needed to run this software. This makes it easy to try image processing approaches before you commit to a sophisticated, and potentially costly, image processing system. Using free image processing software packages can meet the immediate need of running up the learning curve of image processing development, but in most cases it will be inadequate for the actual work. Each of the packages mentioned, Khoros, NIH Image, and UCFImage©, although fairly complete, have

drawbacks and limitations not explored in this article. Also, they do not come with the support that you normally expect when you buy something, which is a feature that deserves careful consideration when you do commit to a commercial software system. In general, software developers want you to use their systems and be happy with them, so look for a strong support structure before you buy and ask to try an evaluation copy before you commit to a program.

HOW TO OBTAIN FREE IMAGING SOFTWARE:

UCFImage is included on diskette in the back of the book *Computer Imaging Recipes in C* (ISBN 0-13-189879-5). Although not technically free (since you have to purchase the book), the UCFImage executable is in the public domain.

NIH Image is available via anonymous FTP from **zippy.nimh.nih.gov** in the **/pub/image** directory. It may also be downloaded from America Online (order an access diskette by calling 1-800-827-6364) or CompuServe (order an access diskette by calling 1-800-848-8199).

Khoros is a huge system that is available via anonymous FTP from various authorized distribution sites under an "open access" policy. One of these is **ftp.khoral.com** in the **pub/khoros** directory. The Khoros open access policy, which is distinct from public domain systems like UCFImage and NIH Image, means that it can be used and modified only for internal use in the organization obtaining it. The user cannot redistribute the system, any derivative works from Khoros, the Khoros documentation or any binaries or libraries which include Khoros object or source code unless the user is a member of the Khoros Consortium and has signed a redistribution license agreement. Documentation regarding how to contact the consortium is included in the software distribution, or you can email a note to **khoros-request@khoros.unm.edu** and they will send you information and pricing.

THE COMPUTER VISION HOMEPAGE

The Computer Vision Homepage at Carnegie Mellon University is one of the most comprehensive sources of information about computer vision research activities on the World Wide Web. Its address is:

http://www.cs.cmu.edu/~cil/vision.html

The site consists of nine major sub-sections:

Vision Groups, Hardware (Research Systems, Commercial Products), Software (Research Code, Image Processing Toolkits, Display Tools, Synthetic Data Generators, Math Toolkits), Demos, Test Images, Conferences, Publications (References, Papers and Proceedings, Books and Tutorials, Journals, Other), General Info (Newsgroups, FAQ's, Archives, Misc.) and Related Links.

REFERENCES

1. Myler, H., "How to choose image processing software for your application," supplement to *Laser Focus World*, September, 1994.

2. Myler, H. and A. Weeks, *The Pocket Handbook of Image Processing Algorithms in C*, Prentice-Hall, Englewood Cliffs, 1993.

3. Myler, H. and A. Weeks, *Computer Imaging Recipes in C*, Prentice-Hall, Englewood Cliffs, 1993.

APPENDIX B
HARDWARE

This appendix focuses on hardware issues that arise in the setting up of machine vision systems. Like software, poor selection of hardware can be costly. It is very important to establish a needs analysis up front before any purchases are made. In the case of high-end imaging, the equipment needed may well run into the hundreds of thousands of dollars. What follows is a checklist format listing (with comments) of hardware consideration issues. The discussions that follow refer to an expanded imaging system (as opposed to just a PC with an imaging display), as shown in Figure 1. The expanded system illustrates all the components of a complete imaging system. In some high-end systems, each of the components is a separate printed circuit card or chassis. Smaller systems combine the frame grabber, frame buffer, digital video processor, and video output controller onto one printed circuit card that fits into the host computer.

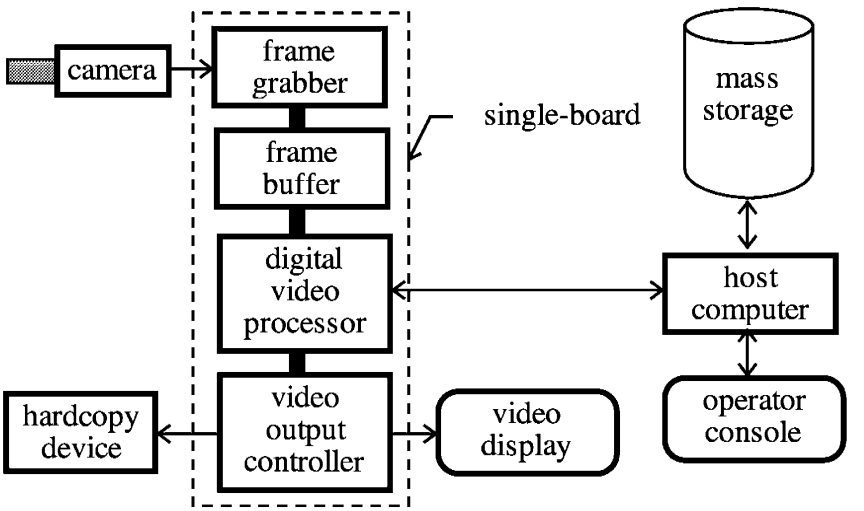


Figure 1. Expanded imaging system

The table below lists the components of the expanded imaging system with a short definition of their functions:

Component	Function
Camera	Captures image data and outputs a video signal (RS-170, NTSC, RGB, PAL, SECAM, etc.) to frame grabber.
Frame Grabber	Digitizes video signal and outputs digital data to frame buffer.
Frame Buffer	Stores image as two-dimensional data array (sometimes called image buffer or image memory).
Digital Video Processor	(DVP) Performs processing on the digital image under direction from the host computer.
Video Output Controller	(VOC) Performs digital to analog (video) conversion and routes video signals.
Hardcopy Device	Video printer.
Video Monitor	Displays image output from DVP.
Host Computer	Orchestrates operation of image system under operator or program control.
Operator Console	Keyboard/monitor allows operator control of computer and system.
Mass Storage Device	Disk to store programs and images.

In some systems hardcopy devices are attached to the computer and process digital data instead of video. Most PC-based systems use the operator console display (computer monitor) to display the image in the frame buffer. High-speed systems for capturing digital movies connect the mass storage device directly to the frame grabber or digital video processor (DVP). The DVP in many systems is a programmable digital signal processor—or the host computer serves as the processor. No matter what the configuration, the components shown above will exist in one form or another in the imaging system.

Cameras: There are a number of video output standards for cameras. Inexpensive monochrome cameras that output RS-170/CCIR (B/W TV quality) video are suitable for many assembly line tasks. High- performance imaging hardware at competitive prices is available for almost any computer platform. For standard color, one must choose either NTSC/PAL/SECAM color TV or RGB cameras. RGB cameras are typically tri-output RS-170 systems. One

must be cautious about sync requirements with any camera. The strategy is to couple the selection of camera to the selection of the imaging system and be sure that the interfaces are compatible.

Cameras for most machine vision tasks can be classified as analog, digital-analog, or digital. An analog camera uses a vidicon to capture an image, which is an electronic tube that outputs an analog signal with the necessary synchronization signals. The camera outputs an analog video signal that must be digitized (by a frame grabber). A digital-analog camera captures the image using a semiconductor chip. Most common is a charge-coupled device, or CCD. The CCD is scanned and an analog video signal is output. A digital camera uses a semiconductor chip but outputs digital data. These cameras do not require digitization and generally have specialized high-speed bus-like interfaces.

The most important parameters to consider when purchasing a camera are the spatial resolution and the pixel sensitivity and wavelength. The spatial resolution is most often expressed as *lines*, which indicates how many rows of pixel data the camera will output. Low-resolution cameras are typically 200 lines, while high-resolution cameras can have 4000 lines. What the pixel actually resolves in terms of spatial resolution will depend on the lens system. Of course, no lens can give more resolution than is actually available at the sensor. As the number of lines increases, so does the cost and the processing demands. The frame grabber will also affect the resolution. If the grabber can only digitize a 256 × 256 pixel image, then a camera with 1024 × 1024 resolution will be overspecified. Digital cameras are simpler in that they output exactly the resolution they are designed to. For example, a 1024 × 1024 digital camera outputs the data for a 1024 × 1024 image. The lens system and the size of the imaging chip in the camera will determine how much of an object can be resolved (see the discussion on lenses below).

The camera wavelength and sensitivity required are dependent primarily on lighting conditions. Most cameras are manufactured for visible light conditions with some sensitivity into the infrared. Filters can be employed to restrict sensitivity and recommendations are readily available from the camera manufacturers.

Lens: Must be specified depending upon the object size to be viewed, the distance from the camera to the object, and the size of the imaging surface. The necessary parameters are shown in Figure 2.

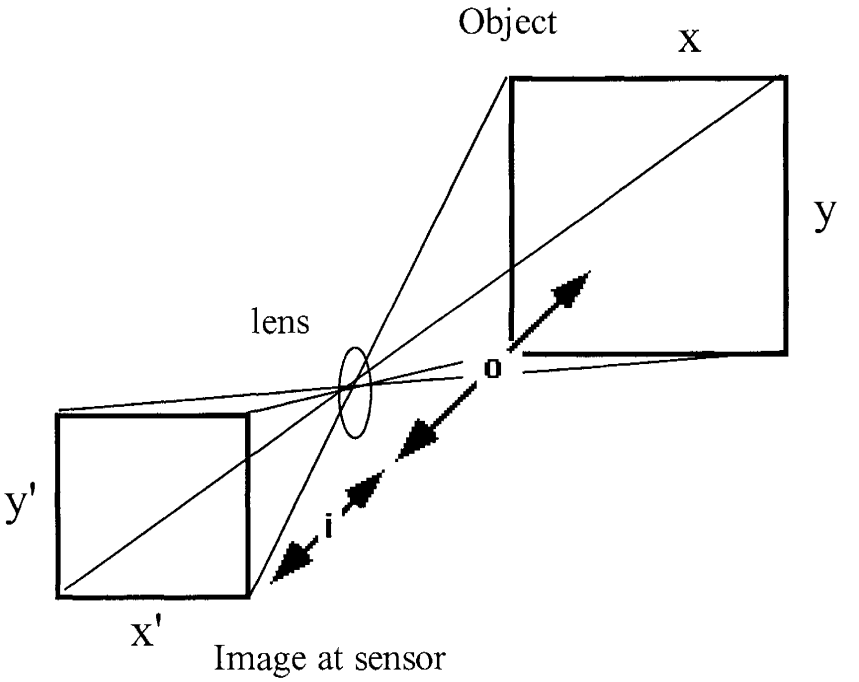


Figure 2. Lens and object parameters.

The resolution per pixel can be computed by dividing the field of view with the number of vertical pixels. The relationships between lens focal length and object/image distances is:

$$\frac{1}{f} = \frac{1}{o} = \frac{1}{i}$$

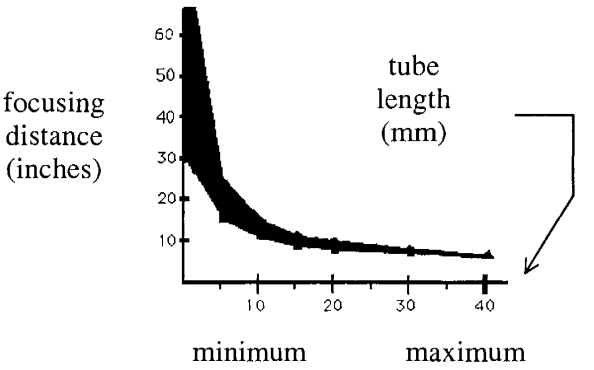
where f = focal length, o = object distance to lens, and i = image distance to lens. Magnification can be computed from:

$$M = \frac{i}{o} = \frac{y'}{y} = \frac{x'}{x}$$

where M = magnification, $y = y$ dimension of object(x) and y' = dimension of image(x).

Extension Tubes: Devices resembling washers or cylinders that fit between the lens and the camera. They increase the image distance (i) to the lens and allow the lens to focus in closer to the object. Charts are available (example

below) that show the effect of focusing distance, FOV, and the resolution of different length extension tubes.



Frame Grabber: This component is used to digitize the image from an analog video signal. The frame grabber may be just a video analog to digital (A/D) converter chip on a single-board system. The input to the simplest configuration monochrome frame grabber is typically an RS-170 BNC or RCA connector with the sync signal overlaid onto the video. More sophisticated systems break out the horizontal and vertical sync signals to allow for greater flexibility in video input. Color frame grabbers can have a BNC or RCA connector for NTSC color (the color TV standard) or individual red, green, and blue (RGB) connectors. Sync for RGB signals can be overlaid onto the green channel or be broken out separately.

No frame grabber is used for a digital camera. Instead, a digital camera uses a digital input port that puts image data directly into the frame buffer or into the host computer memory. The frame grabber can restrict data entering the system. For example, if the camera sensitivity and system design requirements are such that 12 bits per pixel of quantization is required and the frame grabber digitizes to 8 bits, then data is lost.

Frame Buffer: A memory that receives the digitized data from the frame grabber or from the interface for a digital camera. The frame buffer in small systems may be the host computer memory. Typically, however, the frame buffer is very high-speed memory that is directly connected to either the frame grabber or digital camera interface, even in small systems. If no frame buffer exists then image acquisition speed will be dependent on the speed of the host computer bus and memory. Some systems bypass this problem using Direct Memory Access (DMA) from the frame grabber or digital camera interface into computer memory. Once the image is in memory, if the host computer

must access it, then a DMA system will have speed advantages over a separate frame buffer.

Color images are buffered as RGB so three frame buffers are required. Often systems will include a fourth buffer for graphic overlay purposes. High-end systems provide programmable buffers that allow the user to specify spatial resolution and pixel quantization. An important feature is the ability to increase frame buffer size as system requirements change.

Digital Video Processor (DVP): A processor that is specialized for image algorithms. The DVP is typically a set of specialized circuits that have been optimized for histogramming, discrete convolutions, pixel processes and, in some cases, Fourier or cosine transforms. Most DVPs are pipeline processors that perform a sequence of user-programmable operations on streams of pixel data. Some DVPs are parallel processors that divide the image data into segments and process the segments independently. Other DVPs are nothing more than simple serial interfaces to a Digital Signal Processor (DSP) chip, selection of which can incur sophisticated programming demands.

The primary considerations regarding the DVP section of the imaging system are speed and programmability. Almost any DVP configuration will be faster than a host computer used for algorithms, but DVPs can have severe processing restrictions on what algorithms they can perform. Also, programmability can be a major issue. The software provided with the system should be examined carefully for range of capability and ease of implementation. It is very important to have a good system requirement analysis with respect to algorithms needed. This will help avoid the problem of purchasing a system with a DVP that cannot perform the necessary algorithms within the system timing constraints.

Video Output Controller (VOC): Converts digital image data from the frame buffer into analog video for display. The VOC may also perform programmable routing functions to multiple displays from multiple frame buffers. With single board systems the VOC functions may be limited to a single set of NTSC and RGB outputs. Some single board systems have no VOC type outputs and use the computer console display as the output device. In these systems the image data from the frame buffer is transferred to display memory and placed within a window on the computer display.

Single-board System: Combines the frame grabber, frame buffer, digital video processor, and video-output controller onto a single printed circuit board that plugs into the host computer system bus. See the dotted-line region of Figure 1. Single-board systems are typically very cost effective, but can have poor performance and limited capability. The size of the host computer's plug-in cards are what limit single-board systems. Manufacturers have circumvented this problem with multiple-board systems that use a specialized data bus

between them for high-speed image data transfer and a host computer bus for control signals.

Video Display: Typically television monitors that are used to display RS-170 or NTSC analog video data. Higher-end monitors display RGB video. Monitors are specified in terms of diagonal size (inches), dot pitch, or distance from one phosphor dot to the next, and phosphor array size, such as 640×480 or 800×600, etc. Selection of the proper video display can be critical in medical applications where high resolution is generally necessary. For other systems, the display may only be used during system development and programming and so be a less critical consideration.

Hardcopy Device: Used to provide paper output of imagery. This component is more for reporting and archival purposes than it is a critical element of the machine vision system. Hardcopy devices span a wide range of capability and cost. The simplest hardcopy unit is the host computer's printer. Almost all systems include a black and white laser printer and images can be dithered and output to them at minimal cost. Color printers are considerably more expensive, followed by continuous- tone printers that output photographic-quality prints. Cost and image quality are the two driving factors that will determine which printer should be acquired, and vendors will be happy to print a sample image from your data. A secondary consideration is interfaces—be sure that the printer is compatible with the data output that is to be printed.

Host Computer/Operator Console: The computer platform used for the imaging system is possibly the most critical component of all. The host orchestrates the imaging system operation and in complex systems the host is used for programming the imaging algorithms. Larger imaging hardware vendors will have systems that are compatible with multiple platforms. Smaller vendors that address specialized markets or who restrict their markets may build hardware only for specific computers. The selection of which computer to purchase is a major one and the programmability and ease of use of the proposed system should be weighed heavily in the decision to acquire. Software and hardware should be considered concurrently as both will constrain the type and range of algorithms that the system can run as well as performance and efficiency.

Mass Storage: The disk or other media used to store images or image sequences. Various options exist from just sharing the host computer disk a to specialized high-speed, high-volume disks for image sequence processing. Imaging disks attach directly to the imaging system hardware and are capable of real-time acquisition and storage. A simple rule of thumb is that one should always buy as much storage as can be afforded.

Sources for intelligent decisions on imaging hardware abound. One of the best is the numerous imaging conferences and symposia that are held each year

by SPIE (The International Society for Optical Engineering), IEEE (Institute of Electrical and Electronics Engineers) and IS&T (Society for Imaging Science and Technology) where imaging vendors exhibit their wares. It is strongly recommended, however, that software be examined first and algorithm requirements be established. This involves the careful selection of a platform (see Host Computer, above). Once this selection is made, use free imaging software, as discussed in Appendix A, to evaluate needs. Then concentrate on the imaging hardware and software that will be needed for your application. Vendors are more than happy to assist you, but get multiple opinions before you buy.

APPENDIX C
TEN COMMON MISCONCEPTIONS OF MACHINE VISION

This appendix responds to an article that was published in the October 1995 issue of *Advanced Imaging*, a trade journal for electronic imaging professionals, entitled "Ten Common Misconceptions Corrected" by Peter Eggleston. The article offers corrections to misunderstandings that have plagued machine vision and image processing researchers and developers for years. Here the misconceptions have been listed, along with a subset of Eggleston's comments, in italics. The author's comments follow in plain text.

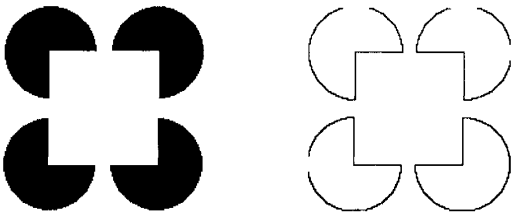
1. If you can't see it, a computer can.
   • *Image processing can be used to enhance data being viewed, but it can't perform magic. If the data was not captured, no amount of processing will "bring it out."*

   This is often a misconception and goes along with the old saw "you don't get anything for nothing." The solution to a potential problem here is to make sure the sensing system captures the data that you need. The reverse of the statement above is not true either. There are cases where you, the human, will not be able to see data but the computer will be able to. This is generally the case where computer data is beyond eight bits of quantization or where very poor contrast masks the presence of objects in shadow.

2. If you can see it, your image analysis software must be able to as well.
   • *Often, you can clearly see objects of interest in your data, but when you apply some image processing to extract them, the objects are not found or single objects are extracted as multiple parts.*

   This misconception states one of the fundamental problems of machine vision, which is getting a machine to see what you do! The sections on thresholding (5.1), segmentation (5.2), and representation (6.1) all discuss aspects of looking at data and then trying to get an algorithm to extract and recognize what you can clearly see. Consider the graphic below, which illustrates a well-known optical illusion. The figure on the left is the illusion—a bright white square is seen over a black

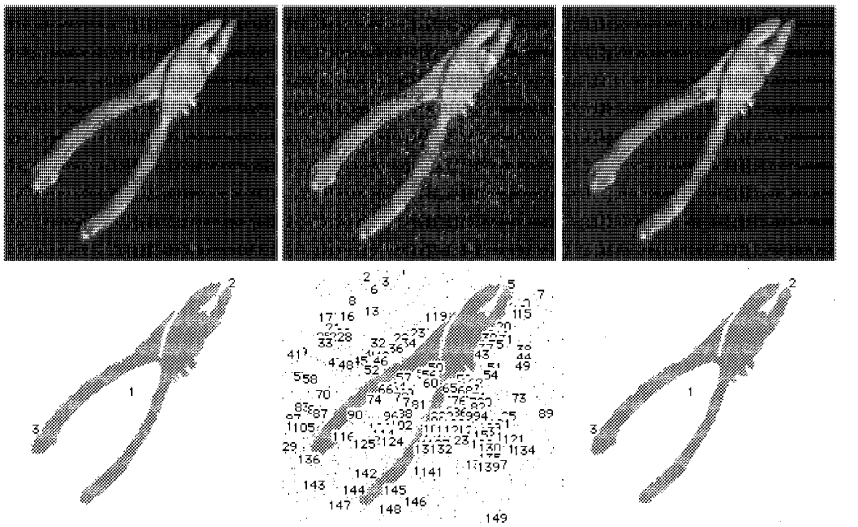
background of circles. Our minds want to see a square, so the visual system completes the edges for us. The figure to the right is the figure after an edge tracing algorithm has been applied—the machine does not see the square and forces the notched circles to be more pronounced. In this example one might argue that the machine is seeing what we should be seeing.



3. Segmentation is a one-step process.
   • *The success of an image segmentation technique often relies on adequate preparation of the data. Proper resolution control, noise suppression, and feature space transformation are important first steps before applying a segmentation technique.*

   In Section 5.2, Segmentation, techniques were evaluated and the follow-up, which is Representation (Section 6.1), was discussed as a means of classifying segmented objects. The textbook examples always show easily-segmented objects, so it is natural to conclude that segmentation is a *one-step process*. Consider the images of pliers on the next page.

   The image of the pliers at the top left was degraded with white pepper noise (top center) and a median filter was applied (top right). Each image was then thresholded to graylevel 180 and a cluster analysis performed with a minimum cluster diameter of 2 pixels. The first cluster analysis (bottom left) found three objects. The second found hundreds and overloaded the system (bottom center) and the last analysis performed on the filtered image found the same three clusters as the first (bottom right).



4. You need a CAD model to use model-based vision.
   • *When they hear the term "model-based vision," many assume the use of computer generated geometric models, otherwise called graphics, to perform matches with image data. However, a model is simply a description or example of an object or system.*

   This misconception arises primarily because of issues of semantics with respect to the meaning of *model-based.* The simplest definition of the term implies any system that stores a representation or model of what it is looking for. The format or nature of that model is open to almost anything. In Section 8.3, Model-Based Vision Systems, we discussed VITREO and PARVO, systems that are model-based on object components called *geons.*

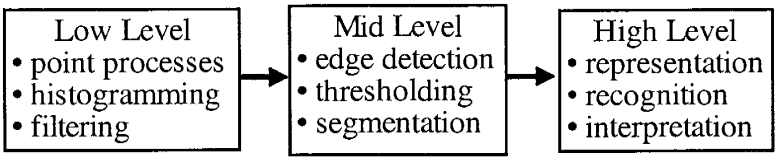
5. Commercial image analysis software is still expensive, even overpriced.
   • *...image analysis is a relatively small market, with most companies only selling a few thousand or even a few hundred copies over the lifetime of a specialized product. Image analysis tools are relatively complex, and very good engineers are needed to develop high-quality tools. And technical support is expensive.*

   Part of the problem has to do with the fact that the machine vision problem is far from being solved (see Section 8.4). When it is (if and when!) then machine vision software will become as inexpensive as word processors or spreadsheets. Until that time, "you get what you pay for" will be the watchword of image processing software systems.

Appendix A discusses machine vision software issues and addresses the process of how to proceed in the acquisition of skills and development software (also note that Appendix B provides a complementary discussion of hardware).

6. Image processing is pixel processing.
   • *...once we start to segment the (image) data and extract attributes of these segments, arrays cease to be the most efficient method for storing and processing this information.*

   Although it is very important to learn methods of "grappling with the pixels," the very process of analyzing an image for machine vision purposes is a process of getting away from pixel data. The entirety of image processing and machine vision can be summarized as three algorithmic stages:



   The input to these three stages is an image. The output of the low level stage is another image, albeit processed. The output of the second stage is better described as a two-dimensional data structure although it may retain features of an image. The output of the final stage is a data structure or description of what was in the input image and where. Of course, the better the description, the more effective the algorithm was.

7. Image processing reduces data.
   • *Eggleston quotes John C. Russ from* The Image Processing Handbook *(see bibliography): "Image processing does not reduce the amount of data, it just rearranges it. Its goal is to give more meaning to the data."*

   It is very important to understand that one need never lose the original image data. Many algorithms process the image and reduce data, but these processes need not be *in place* such that original data is overwritten. It is sometimes misunderstood that once processing takes place one cannot also use the original information in the ensemble of processes leading to conclusions about image content.

   The argument Eggleston and Russ are making is that, if anything, an image processing operation may create more data. Now, this is not to say that more information has been created—that cannot happen. What

it does say is that a presentation of the data to reveal information present has been accomplished.

8. Color processing means processing red, green, and blue channels.
   • *Color is a characteristic of grabbed image data. Red, green and blue (RGB) are merely samplings of the color space.*

   Color processing is often confusing and gives rise to misconceptions because the greater bulk of image processing has taken place on monochromatic images because of hardware costs. With the advance of technology, color processing has become cost manageable both from a research perspective and industrial implementation. Nevertheless, one must understand that an RGB image is a multisensor representation. There are three images involved and their independent processing gives credence to the above misconception. Processing RGB imagery is a multisensor fusion process. When RGB is transformed into other color spaces, such as hue saturation intensity (HSI) or C-Y, then the processing is color processing.

9. You can tell the power of an image processing system by the number of operations it has.
   • *Since it can be difficult to get a good understanding of a product's capabilities from the marketing literature, counting up the number of operations provided may seem like a good approach to determining the power or value of a software package...*

   Well, this misconception arises in many different fields. For example, keep in mind the day you discovered that a dime was worth more than a nickel! Eggleston illuminates two important points: (1) the algorithms supplied in an imaging system may not be the ones you need, and (2) some algorithms are packaged differently between different systems. Therefore, number of algorithms is not a good measure of a system. Also, some systems are designed to address specific applications, such as medical or aerospace. A good measure of a system is who uses it and the successes that have been achieved. These are well documented in the literature.

10. The ideal image analysis tool could be used by anybody.
    • *Even if we limit it (image analysis tools) to anybody who has images to analyze the reality is that we are all not the same; we all have different experiences, training, expectations, prejudices and skill levels.*

    Let's at least say that it certainly hasn't been developed yet! This misconception stresses the importance of careful evaluation and the understanding that image processing and machine vision are

non-trivial areas of development—whether research or industrial application. The key to overcoming any bias you may have regarding this misconception is education.

ANNOTATED BIBLIOGRAPHY

The books listed in this bibliography range from the classic to the most recently published on image processing and machine vision. They are listed in order of publication year and include notes regarding their content.

Digital Image Processing
K. Castleman, Prentice Hall, 1996

Senior/graduate level imaging text with problems and projects. Excellent coverage with signal analysis, wavelets, optics, 3-D imaging and machine vision topics.

Two-Dimensional Imaging
R. C. Bracewell, Prentice Hall, 1995

Senior/graduate level imaging text with problems. Heavy emphasis on signal analysis. Chapter on synthetic aperture radar.

Pocket Handbook of Image Processing Algorithms in C
H. R. Myler and A. R. Weeks, Prentice Hall, 1993

Reference for a wide range of imaging algorithms. Tested C code for algorithms with cross-reference by class, subject, and algorithm name.

Computer Imaging Recipes in C
H. R. Myler and A. R. Weeks, Prentice Hall, 1993

Senior level imaging reference text with examples and emphasis on computer implementation. Book includes diskette with UCFImage© image processing software (DOS).

Digital Image Processing, 2nd Edition
R .C. Gonzalez and R. E. Woods, Addison-Wesley, 1992

Classic and widely used senior/graduate level imaging text with problems. Signal processing emphasis with good coverage of segmentation, representation, and recognition techniques.

Computer and Robot Vision, Volumes I and II
R. M. Haralick and L. G. Shapiro, Addison-Wesley, 1992

Graduate level machine vision text with problems. Extensive coverage of all aspects of machine vision.

The Image Processing Handbook
J. C. Russ, CRC Press, 1992

Reference imaging text with large number of continuous-tone images. Excellent coverage of color image processing and numerous examples of algorithms throughout book.

Vision, Instruction and Action
D. Chapman, MIT Press, 1991
MIT doctoral thesis in Artificial Intelligence. Describes a sophisticated integrated system that takes instruction, interprets its environment visually and plays video games on its own. Provides an implementation of a unified visual architecture in the machine.

Artificial Vision for Mobile Robots
N. Ayache, MIT Press, 1991
Research monograph on research into 2 and 3-D robot vision at INRIA. Complete coverage of 3-D vision system algorithms for sensing, representation, interpretation, and guidance.

Machine Vision and Digital Image Processing Fundamentals
L. Galbiati, Prentice Hall, 1990
Senior/vocational-level imaging text with problems. Good coverage of basic techniques with system design examples. Chapter on barcode analysis.

Nonlinear Digital Filters
I. Pitas and A. N. Venetsanopoulous, Kluwer Academic, 1990
Graduate level imaging text and reference. Extensive and thorough coverage of nonlinear digital filters. Performance evaluation of various filters described.

Digital Image Processing and Computer Vision
R. J. Schalkoff, John Wiley & Sons, 1989
Graduate level imaging text and reference. Strong math emphasis with artificial intelligence approaches to machine vision.

Digital Image Processing
W.K. Pratt, John Wiley & Sons, 1978
Third Edition, 1989
Classic graduate level imaging text and reference with problems. Extensive and thorough coverage of all aspects of image processing with emphasis on stochastic modeling.

The IR Handbook
W. Wolfe and G. Zeiss, eds., Office of Naval Research, U. S. Navy
3rd Printing, 1989
Classic reference book with sections on imaging and tracking systems.

Structured Matrix Image Processing
E. R. Dougherty and C. R. Giardina, Prentice Hall, 1987
Graduate level imaging text with problems. Matrix approach to imaging with strong math emphasis. Extensive coverage of morphological and topological operations.

Intelligence: The Eye, the Brain and the Computer
M. A. Fischler and O. Firshein, Addison-Wesley, 1987
Graduate level imaging text and reference. Extensive and thorough coverage of non-linear digital filters. Performance evaluation of various filters described.

Fundamentals of Interactive Computer Graphics
J. D. Foley and A. Van Dam, Addison-Wesley, 1984
Graduate level computer graphics text and reference. Extensive and thorough coverage of fundamentals of advanced computer graphics algorithms.

Computer Vision
D. H. Ballard and C. M. Brown, Prentice-Hall, 1982
Classic graduate level machine vision text and reference with problems. Math intensive with some emphasis on medical imagery. Has become somewhat dated and superseded by Haralick and Shapiro (see above).

Machine Perception
R. Nevatia, Prentice-Hall, 1982
Senior/graduate level machine vision text. Very well written and easy to follow.

Computer Image Processing and Recognition
E.L.Hall, Academic Press, 1979
Graduate level imaging text with problems. Somewhat math intensive with emphasis on photometric (physics-based) imaging.

Pattern Recognition and Scene Analysis
R. O. Duda and P. E. Hart, John Wiley & Sons, 1978
Classic graduate level machine vision text and reference with problems. Merged pattern recognition principles with machine vision techniques.

Digital Picture Processing, Vols. I and II
A. Rosenfeld & A. C. Kak, Academic Press, 1976
Graduate level imaging and machine vision texts and references with problems. Very math intensive with emphasis on images represented as stochastic processes. Volume I is image processing and Volume II concentrates on machine vision algorithms.

An Introduction to Morphological Image Processing
E. R. Dougherty, SPIE Press, 1993.
A general treatment of morphological image processing written for the practicing engineer. This book covers the classical techniques of morphological processing in an easy to read and understand fashion.

INDEX

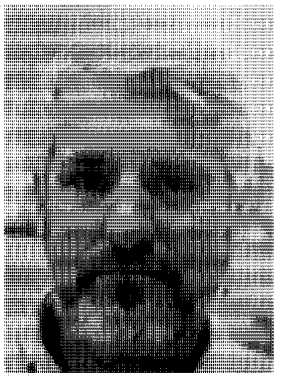
R

S

T

V

W



Harley R. Myler is a Professor in the Department of Electrical and Computer Engineering at the University of Central Florida in Orlando. Dr. Myler did his graduate work at the Electronic Vision Analysis Laboratory at the New Mexico State University, earning the MSEE in 1981 and the doctorate in 1985. He is currently the Director of the Machine Intelligence and Imaging Laboratory at UCF and has published over 30 articles and four books in the areas of imaging science and engineering, computer programming, and architecture and engineering education. Dr. Myler is a member of SPIE, a senior member of the IEEE, a Tau Beta Pi eminent engineer and a member of Eta Kappa Nu.