MACHINE VISION FOR ROBOTICS AND INSPECTION

MACHINE VISION SYSTEMS IN CONTEXT

Vision in Nature, Computers, and other Machines

What a Machine Vision System Is and Is Not. Human vision has always been and still remains the single most important sensing facility for the manufacturing industry. Vision is inherently clean, safe, and hygenic since it does not rely on physical contact between the inspector and whatever object he or she is currently examining. Vision is also extremely versatile and is capable of detecting very subtle changes of shape, size, texture, shade, and color. A person is able to resolve highly complex and often ambiguous scenes and is almost always able to make appropriate safe decisions about previously unseen events. While human vision remains dominant in manufacturing industry, there is a strong research effort aimed at automating visual inspection and other related tasks.

This article explains how machines can be provided with visual sensing, thereby enabling them to perform a wide range of industrial tasks: inspecting, counting, grading, sorting, matching, locating, guiding, recognizing, identifying,

J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*.

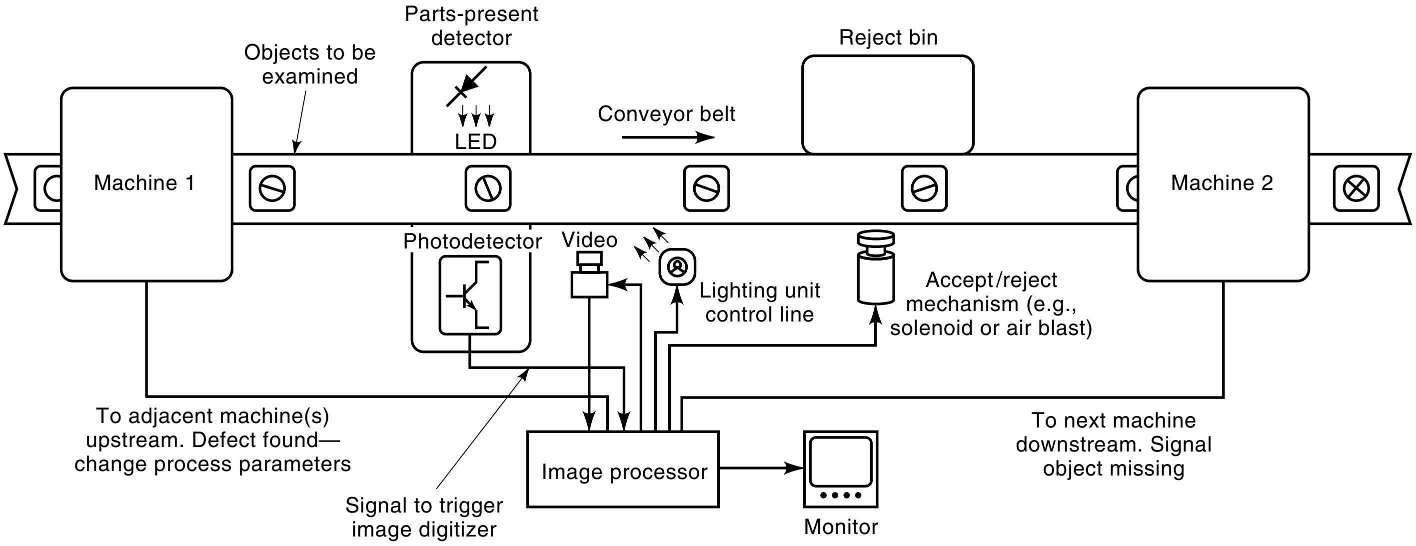

Figure 1. MV system block diagram for on-line product inspection and/or process control.

reading, classifying, verifying, measuring, controlling, calibrating, and monitoring. Machine vision (MV) systems can be used to examine raw materials, feedstock, tools, processes, partially machined and finished products, coatings, labels, packaging, transport systems, waste materials, and effluent.

The block diagram of an archetypal MV system is shown in Fig. 1, in which its multidisciplinary nature is immediately apparent. This is the single most important fact about MV systems; they are not, as might be imagined, simply standard computers with an image-grabber board, a video camera, and some appropriate software. Such an arrangement is likely to be far too slow for use in many practical applications, such as on-line inspection. In any case, we certainly need to add suitable mechanical handling facilities, lighting, and optics to form a complete MV system. However, for the moment, the concept of the image-processing engine as a computer that has been interfaced to a video camera provides a convenient mental picture with which we can begin our discussion (Fig. 2).

Industrial MV is not concerned with understanding, or emulating, either human or animal vision systems. Indeed, the huge differences that exist between human and animal vision on the one hand and MV on the other are so great that we must put aside all thoughts of how we *think* we see the world. Almost without exception, everyone who enjoys good eyesight is, in their own opinion, an expert on vision. However, popu-

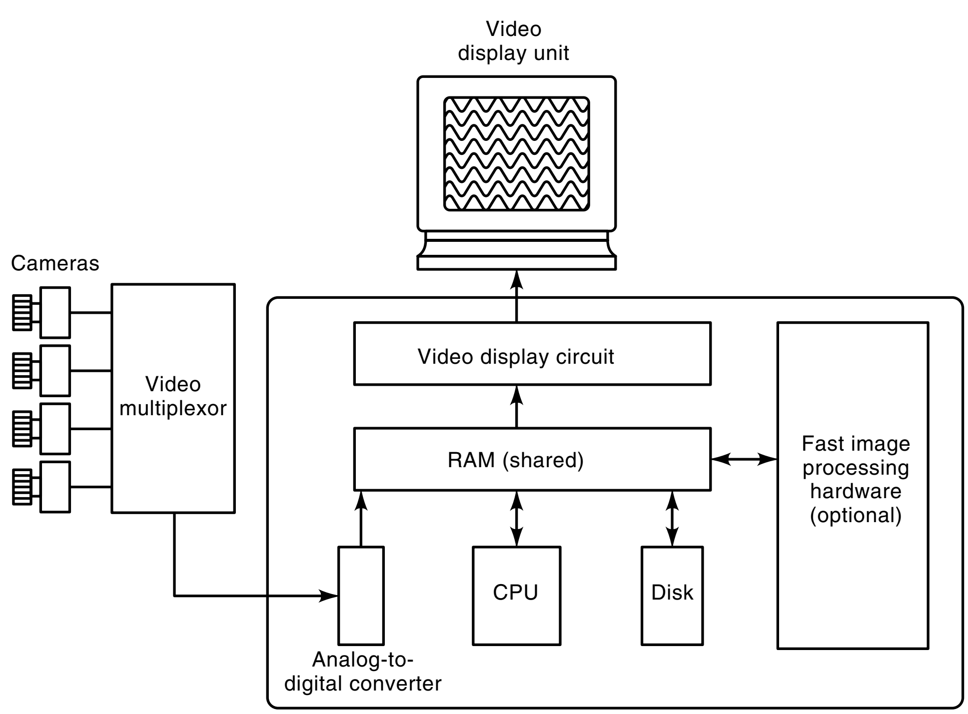

Figure 2. Naive view of the image-processing subsystem as a computer with a video input port. For the sake of simplicity, the synchronization and control signal paths are not shown. Notice that there is a possibility of adding a high-speed image-processing hardware unit.

lar belief does not necessarily coincide with the truth, and this idea is certainly not justifiable on close scrutiny. Carefully controlled psychological experimentation has shown quite clearly that people do not see things exactly as they think they do. As a result, models of the human visual system based upon introspective self-analysis are unable to predict exactly how a person will respond to a given visual stimulus. Natural vision is extremely subtle and cannot yet be emulated by a machine. As evidence of this, consider just one seemingly straightforward task: recognizing an "attractive-looking" person. To a human being with good sight, this task is trivial but it has so far proved to be totally impossible for a machine, as has that of recognizing a given person's sex.

Human Vision is Too Complex to Emulate in a Machine. Vision accounts for about 70% of the data flowing into the human brain. This occurs at an enormous rate: approximately 10^{11} bit/s. The human visual system has been refined during many eons of evolutionary development and has resulted in an extraordinarily efficient design, involving many billions of nerve cells (roughly 10^{13} neurons for a young adult). No machine that we can conceive of building in the foreseeable future can match the rich connectivity that exists between the computing elements (individual neurons) in the central nervous system of a human being. Our knowledge about natural vision, though growing rapidly, is still very rudimentary, compared with what remains to be discovered about the brain and its operation. Even the most powerful modern multiprocessor computer does not have the processing power to simulate the nervous system of an insect, even if we knew how to program it to do so. Many years ago John von Neumann postulated that the simplest description of the human brain may be the human brain itself. So far, this conjecture remains unresolved. The conclusion to which we are drawn is that MV systems must be designed on engineering principles and not by slavishly following the precedents found in biological systems.

Understanding Relies on Past Experience. One of the most powerful features of human vision is a person's ability to relate what he or she sees to his or her previous experience. Providing a link between seeing and understanding is one of the major challenges for designers of MV and computer vision systems. For a machine, an image is usually represented as an array, containing a large number of integers. (See the section entitled "Image Processing for Machine Vision.") Understanding an image represented in this form in a machine may be likened to a person looking at a book written in a foreign language. The reader receives all of the visual information necessary to read the text but does not have the relevant syntactic and semantic knowledge needed to extract its meaning. A machine has to be given this knowledge implicitly or be supplied with appropriate learning rules, enabling it to extract perceptually useful information from the images it sees. It is relatively easy to fool and confuse human visual perception: optical illusions, numerous psychological tests, and the work of various artists all provide ample evidence of this. (The effects can be so powerful that the observer feels dizzy and nauseous.) It is, of course, both unreasonable and unnecessary to expect MV systems to exhibit this particular type of aberrant behavior.

MV Systems Cannot Operate Properly in Uncontrolled Lighting. People are frequently able to cope with previously unseen situations, whereas a machine vision system may not be able to do so. A notable example, all too often encountered on the factory floor, occurs when the scene being viewed is illuminated naturally. Sunlight may fall on a certain spot only very infrequently, at a certain time of day or year. For the rest of the time, that point may be relatively dark, despite the provision of artificial lighting. It is important to realize that artificial environmental lighting is also highly variable; shadows may be cast by workers standing nearby, or lamp filaments may fuse. In addition, the light output changes, as lamps age and as dust settles on them. For these reasons, it is vitally important that uncontrolled lighting does not does impinge on the image-aquisition subsystem. If it does, dangerous products may pass undetected, good product may be erroneously rejected, or the vision system may halt the manufacturing process. It is usually far easier to control the lighting locally than to try to build unnecessarily complicated machines that sales personnel can then claim will operate whatever the lighting conditions. Here, more than anywhere else, we see the utter folly of trying to apply the lessons gained by studying human vision to machine vision, without a great deal of thought and analysis.

MV is Systems Engineering

In the previous section, we argued that MV systems should operate in carefully controlled lighting conditions. Of course, human beings do not need to do so and are able to read, for example, in moonlight or brilliant sunlight. An essential feature of MV is that it is concerned with much more than the process of "seeing." For this reason, we must also be careful to distinguish between *machine vision* and *computer vision*. The latter is a branch of *computer science* and is concerned solely with the computational techniques needed for analyzing images. The "starting point" for studying computer vision is a digitized image, inside a computer. On the other hand, MV is an aspect of *systems engineering* and is concerned with a much wider range of technical issues:

1. Object or material transport (to and from the camera)
2. Image acquisition (lighting, optics, optical processing)
3. Image sensing (i.e., the camera or other radiation-sensing device)
4. Image processing (this takes place within a machine, which may or may not include a device that is immediately recognizable as a computer)
5. Software and computational procedures for image manipulation
6. Control of external electromechanical devices
7. User interface (both input and output)

The starting point for MV is typically a piece of metal, plastic, wood, or glass that is to be examined.

Applications Constraints. An essential feature of MV is that the system designer must take into account the numerous and very diverse contraints imposed by the application, in order to achieve an economic and effective design. Computer vision is not limited in this way. To emphasize the importance of this distinction between machine and computer vision, consider the task of inspecting piece parts that are moved past the camera using a continuously moving conveyor. Here are a few of the arguments that a systems designer must ponder in this situation:

1. While we can use a standard array camera, it is necessary to use stroboscopic illumination. Flashing lamps can induce epileptic fits and migraine attacks in susceptible individuals, so good light screening is essential. Instead, a high-sensitivity camera could be used in association with a fast liquid-crystal display (LCD) shutter. This does not present a health hazard but is not quite so versatile.
2. A line-scan camera is ideal, but the speed of the conveyor belt must be very carefully controlled and measured, so that the camera can be continuously recalibrated. The arrival of an object in front of the camera is usually detected by a photocell. However, it must be borne in mind that the light source associated with that photocell [often an infrared light-emitting diode (LED) that is invisible to an engineer] illuminates the piece part too and hence may well reduce the image contrast.
3. A certain architecture for the image-processing system (i.e., a concurrent processor) is well suited to inspecting objects on a conveyor belt. Parts are effectively "stored" on the conveyor belt, while the image processor calculates the result for each one in turn. In this way, it is possible to accommodate a high latency while maintaining a high throughput rate.

Inspection. As indicated before, MV systems are being used in a very wide range of industrial applications. One of the earliest and still one of the most important of these is the inspection of manufactured products. Detecting faults in piece parts and continuous materials made in strip or web form are two of the prime areas for applying MV systems. Faults may take a variety of forms: type, shape, size, color, scratches, cracks, splits, pitting (due to trapped air bubbles, or other inclusions), swarf, mislabeling, chemical contamination, oil, rust, oxidation, staining, water damage, heat damage, misassembly, foreign bodies (adhering to the surface or "machined in"), missing features (e.g., holes, cables, springs, bolts, washers, nuts, and rivets), material defects, incomplete machining (e.g., untapped hole, missing chamfer, no final polishing, and no slot on a screw), and missing or incomplete coating. The word *inspection* is used in two ways: to refer to the specific task of detecting faults such as these, and in the more general sense to include a range of applications such as counting, grading, sorting, calibrating, locating, identifying, and recognizing. In fact, the term *automated visual inspection* is often used to cover industrial applications of MV other than robot vision. However, there is no clear distinction between automated visual inspection and robot vision, since many inspection tasks require parts manipulation and many object-handling applications also require identification and verification, before deciding that it is safe to move the robot and calculating how to move it.

Robot Vision. For our purposes, the term *robot* can be applied to any machine that provides the ability to move its end effector (e.g., a gripper, tool, or camera), under computer control, to a given point in two- or three-dimensional space. It may also be possible to control the orientation of the end effector. According to this definition, the following machines qualify to be called robots: numerically controlled milling machine [3 degrees of freedom (dof)], lathe (2 dof), drill ($2\frac{1}{2}$ dof), electronic component insertion machine ($3\frac{1}{2}$ dof), (X, Y, Z, θ) table (4 dof), graph plotter ($2\frac{1}{2}$ dof), Selective Compliance Automatic Robot Arm (SCARA) arm ($2\frac{1}{2}$ dof), gantry robot (4 dof), serpentine arm (many dof), articulated arm (up to 6 dof), autonomously guided vehicle (3 dof), crane (3 dof), and hoist (4 dof).

The first industrial robots were "blind slaves" and proved to be useful in repetitive tasks, in which the form and posture of the workpiece is predictable. However, they were severely limited in their total inability to cope with new and unexpected situations. While the accuracy of the robot may be very high, it may be expected to operate in an environment in which the parts delivered to it are in a unknown position or posture or may have a variable size or form. Vision is, of course, ideal for locating objects, prior to directing the robot to pick them up. Without appropriate sensing of its environment, a robot can be dangerous; it can damage itself, other equipment, the workpiece, and personnel nearby. Vision provides one of the prime sensing methods, for obvious reasons. A visually guided robot can "look ahead" and determine that it is safe to move before committing itself to what would be a dangerous maneuver. An intelligent visually guided robot can, for example, pick up an article that has fallen at random onto a table, even if it has not seen the article before. Without the use of vision, that would be impossible.

Coordinate Systems and Calibration. Prior to using a robot vision system, it is essential that it is calibrated so that the position of an image feature located by the camera can be related to the position of the robot arm, and vice versa. The exact nature of the calibration procedure is specific to each case, since many factors affect the process: arm geometry, camera location, geometric distortion of the image (due to limitations of the optical subsystem, or the camera's spatial nonlinearity), arm/end-effector geometry, lighting, features visible in the background, etc. The purpose of the calibration process is to derive suitable values of those parameters needed by the equations that relate the robot arm position to observable points in two-dimensional (2-D) space, as seen by a camera. Using more than one camera, we can, of course, locate a point, such as a given feature on a workpiece or the tip of a robot's tool, in 3-D space.

It is useful to note that three convenient methods exist for making a given point on the robot arm stand out clearly, so that the vision system can identify it easily and then locate it accurately:

1. Fit a small patch of retroreflective material to the arm. A lamp placed very close to the camera will make this patch appear to shine very brightly.
2. Fit a small beacon to the arm consisting of a single optical fiber.
3. Follow method 2 but use an LED instead.

In all three cases, the lamp or LED used to highlight the calibration beacon can be switched off during normal use. However, it should be understood that the primary task during calibration is to relate the *end effector* position to the camera, not some arbitrarily placed target on the robot arm. (Imagine trying to perform some complex manipulation while being able to see your wrists but not your fingertips.) In many cases, it is not possible for the camera to view the end effector directly. However, various schemes have been devised in which the *actions* of the end effector are made visible to the camera. For example, the robot may position a marker and then move right out of the camera's field of view, so that the vision system can locate the marker. In this way, it is possible for the vision system to find out where the robot *was,* not where it *is.* Similarly, a robot drilling machine could be ordered to drill a hole in a position defined according to its own coordinate system, then move away, so that the vision system can locate the hole. For the purposes of calibration, information about the past position of the robot is just as useful as knowing where it is now.

A robot may use any one of the standard coordinate systems: Cartesian (e.g., gantry type robot), polar, or cylindrical coordinates (articulated arm). On the other hand, SCARA, serpentine, and certain other types of robots use no established coordinate system at all. In these later cases, the physical construction of the robot arm is such that the controllable physical parameters (i.e., angular positions of rotary joints and/or the linear position of a carriage along a slide-way) may not relate directly to the familiar Cartesian coordinates. A Cartesian coordinate system is usually preferred by a person since (X, Y, Z) parameters relating to the position of a landmark point on the robot can be determined using nothing more complicated than a ruler. Most camera systems measure position in terms of two orthogonal (i.e., Cartesian) parameters.

For any robot-vision system, it is vitally important that the position of the end effector can be related accurately to the camera's field of view. Indeed, this is essential for the safe operation of the robot. For this to be accomplished, it is important that the system be calibrated before use and checked regularly afterwards, during operation. Recalibration can make good use of the tiny bright beacons described earlier. Failure to check the system calibration regularly can lead to the robot, camera, lights, or workpiece being damaged by the robot not being where it is supposed to be.

Elements of a Vision System

Image Aquisition. A digital image is formed by the interaction of eight distinct physical entities:

1. Power supply for the light source
2. Light source (lamps, LEDs, laser, etc.)
3. Object being examined (including any coatings and any surface contamination)
4. Optics (including the camera's face plate, environmental-protection window, and the main camera lens)
5. Atmosphere
6. Photodetector (array)
7. Analog preprocessing of the video signal
8. Digitization circuitry

Noise can enter the system in various ways, at points 1 to 7. For a vision system, airborne fumes, smoke, spray, and dust, together with surface contamination (oil, water, stains, rust, etc.) are major sources of "noise." In addition, stray light from reflections of nearby objects, environmental lighting, sunlight, etc., represents another form of noise. Modeling the image acquisition subsystem is highly complex and prone to certain inherent difficulties, in view of the fact that we cannot properly predict the optical effects of surface dirt, staining, rust, airborne dust, fumes, and other forms of contamination. It is also virtually impossible to model the effects of stray light and shadows on the image quality, since they are both totally unpredictable. For this reason, very great care must be taken to eliminate ambient light from the camera's field of view. It is still probably more cost-effective to design the image-acquisition subsystem empirically, using traditional engineering principles, rather than the limited set of computer-based tools available.

Choosing the Right Lighting Configuration. The importance of the lighting and viewing subsystem for MV cannot be overemphasized. An ill-designed image-acquisition subsystem can often transform what should be a very straightforward image-processing task into one that is impossibly difficult. For example, brightly polished, embossed silver and molded or cut glass both place special demands on the design of the lighting system. In the former case, the main problem is caused by local highlights, due to glinting (specular reflection), which can cause optical overloading of the camera. Clear glassware can also produce highlights due to glinting from the surface but the main cause of optical overloading is a combination of refraction and internal reflection. Several solutions exist for both of these problems, but it should be understood that there are many other "difficult" materials and shapes for objects requiring inspection. Choosing the optimal lighting and viewing technique is often far from easy. This is one of the most critical areas of MV and deserves full attention by the system design engineer. However, it is not possible in an article of this length to review all of the possible lighting and viewing methods that have been proposed for MV and the reader is referred elsewhere for this. A catalog of about 180 different lighting and viewing techniques has been compiled and is available in the form of an interactive program. It is also available via the World Wide Web (1).

Optics. There are many ways to use optics to enhance the image presented to the camera. A large range of optical components is available and may be used to good effect in many applications. For example, multilayer (interference) filters can create some remarkable effects when viewing multicolored objects by providing a large increase in image contrast. Polarizers can make stress in transparent materials visible and can also be used to reduce the effects of glinting on bright shiny objects. Mirrors and prisms can be used to enable a single camera to look in several directions at once. Fiber-optic devices allow cameras to view in very restricted spaces.

Needle scopes as small as 1.2 mm diameter can generate images of excellent quality, while coherent fiber-optic bundles of a similar size and up to 3 m long are available. Modern molded plastic lenses are much cheaper and lighter than their glass counterparts and can be machined easily to create special effects. A wide range of other optical devices have been proposed for machine vision, including slits, gratings, Fresnel lenses, curved mirrors, cylindrical lenses, quarter-wave plates, holographic lenses, prisms made from birefringent materials, and spatially modulated (liquid crystal) light valves. The important point to note here is that the optical subsystem requires just as much attention by the vision system designer as any other part, since the image quality can greatly be enhanced at a relatively low cost.

Camera or Image Sensor. Although human sight is obviously associated with the eyes, these are merely *image sensors* and therefore form only part of the human vision system. Of course, the brain is intrinsically involved in human vision too. The equivalent component to the eye in MV is a video camera, although this is a very crude instrument in comparison. The human eye is an exquisitely sensitive and versatile organ. It has a dynamic range of about $10^{14}:1$, whereas the best cameras cannot yet do better than about $10^{8}:1$. In addition, the human eye can discriminate far more colors and shades of intensity than the best cameras currently available. Moreover, the eye automatically compensates for spatial and temporal variations in illumination in a most remarkable manner. The act of watching people swimming provides ample evidence of the ability of the eye and brain to compensate for intense local variations in lighting. No video camera can yet match this capability.

Animals have evolved a variety of types of eye: fish, insects, nautiluses, and pit vipers provide notable examples. Again, MV does not set out to emulate them. However, MV systems can use one of the many novel forms of image sensor that have no biological counterpart. These include circular-, line-, radial-, spiral-, and random-scan photodetector arrays, multispectral sensors, X ray, γ-ray, microwave and ultrasonic imagers, laser scanners, and strobed cameras. UV- and IR-sensitive cameras are also available, although these do have biological parallels. Cameras can be placed in situations in which animals would perish due to radioactivity, excessively high or low temperatures, toxic atmospheres, or in which there is a high risk of infection by microorganisms. A camera can be made completely sterile, whereas a person, however well protected, cannot. Cameras can now outperform the human eye in very low light levels, being able to capture individual photons. When they are fitted with specialized optics, cameras can also perform some remarkable feats. For example, satellite imagers are now able to read a newspaper headline from a space vehicle in low earth orbit. The important point to note is that eyes and video cameras have quite different strengths and weaknesses and we must be careful not to expect a camera to see everything that a person can.

Some cameras can be controlled from a computer, via its serial (RS-232) port. Pan, tilt, zoom, video standard, automatic gain control (AGC), integration time, shutter-speed, white level, and gain are among those parameters that can be controlled via software.

Although they have many disadvantages, tube cameras still offer some benefits over solid-state sensors, which are by far the most popular types of image sensor used in practice. Tube cameras are usually preferred when special scan patterns are required, extended spectral response is needed, or when extremely high sensitivity to light is needed. Tube cameras can sense far-infrared and ultraviolet radiation, as well as X rays. They are also used in areas in which there is a high level of radioactivity, since solid-state sensors are destroyed very quickly by neutron bombardment.

Image Digitization. Circuits for digitizing standard (RS-170 and CCIR/RS-320) video signals are readily available as plug-in cards for the most popular computers. They are able to accommodate both monochrome and color signals. Some computers intended for multimedia use (e.g., Apple Macintosh AV and Silicon Graphics machines) are supplied with video inputs as standard.

Commercially available devices also exist for digitizing data from a line-scan camera or flying-spot (laser) scanner. Nonstandard cameras, such as those generating a very-high-resolution image or with special scan patterns, may require specially designed circuitry. However, nowadays most camera manufacturers are well aware of the need to provide an interface to a computer and supply some form of device for this.

There exists a wide and rapidly growing range of cameras that have been designed to connect directly to a computer's serial or small computer systems interface (SCSI) port. Many of these are designed to be used free-standing, like a film camera, except that the internal storage medium is electronic. Of course, for MV, we require a camera to operate on-line, permanently connected to the image digitiser.

Another class of image digitizer is used to generate a bit stream for further processing, by dedicated high-speed electronic hardware, rather than a computer. (A computer is almost invariably involved as a controller for the image-processing boards and, if necessary, to perform high-level image-processing tasks. This term is explained later.) However, to maintain high processing speed, the raw image is not loaded into the computer.

Image Preprocessing (Low-Level Vision). Data are generated by a video camera at quite a high rate. For example, scanning a 512×512 pixel image, at a rate of 25 frames/s, generates data at 6.6 Mbyte/s. Sustaining data analysis at such a high rate is impossible for the present generation of general-purpose computers. For this reason, it is often necessary to employ some analog and/or digital hardware to process the digitized video signal. The aim of this is to reduce the data rate while, if possible, improving the contrast of features that we are expected to isolate and measure.

Detecting cracks in glassware may be used to illustrate this point. A typical algorithm is as follows:

1. Digitize an image from the camera. (Call it Q.)
2. Save Q in a temporary store.
3. Blur image Q. (Call this result Q^*.)
4. Read image Q back again.
5. Subtract image Q from Q^*. (Call this result W.)
6. Threshold image W. (Call this binary image B.)
7. Select and measure the largest blob in image B.
8. Decide on the basis of these measurements whether a significant crack is visible.

Steps 1 to 6 can be performed in commercially available digital hardware. Certain measurements (step 7) may also be amenable to implementation in fast dedicated electronic hardware, while other more complicated parameters may require the use of a general-purpose computer. Some fairly naive procedure may be adequate for decision making (step 8). In this case, we need not involve a computer at all. On the other hand, a much more sophisticated approach may be required at step 8 and this falls under the umbrella of *high-level* processing. A brief review of the more important low-level image-processing operators used in MV is given later.

Image Analysis (High-Level Vision). The term *high-level vision* covers those image-processing operations needed to understand the significance and interrelationships of the various features in an image. On the other hand, *low-level vision* typically includes tasks such as detecting and measuring image features. In many vision applications, it is necessary to create, manipulate, and analyze an *abstract, parametric,* or *symbolic* representation of the object or scene being examined. In this case, high-level image analysis is needed and relies on techniques that would typically be described as belonging to artificial intelligence. We need to distinguish between low- and high-level vision because they depend upon quite different concepts and are best expressed in different types of computer language.

In order to understand the nature and function of high-level vision, we shall consider the task of recognizing a standard table place setting. (See Fig. 3.)

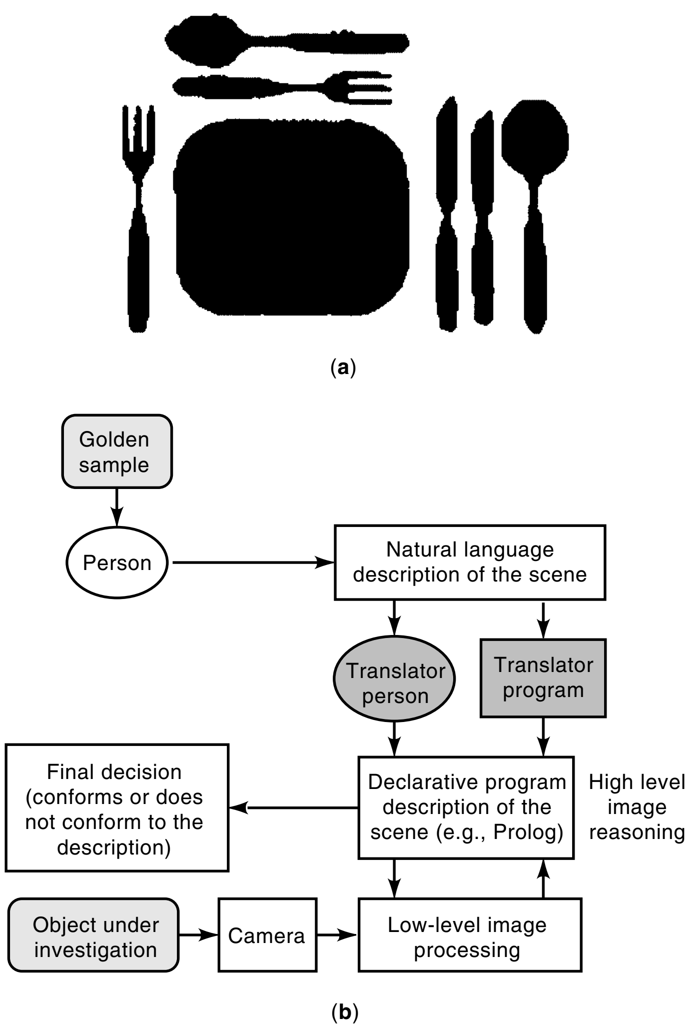

Figure 3. Table place setting. (a) Standard layout. (b) Understanding natural language descriptions of a scene.

A person might typically describe this arrangement in English in the following way:

The soup spoon is to the right of the small knife.
The small knife is to the right of the dinner knife.
The dinner knife is to the right of the mat.
The dinner fork is to the left of the mat.
The desert fork is below the desert spoon.
The desert fork is above the mat.

Verifying that a given arrangement of table cutlery and flatware conforms to such a specification presupposes that we can recognize entities such as *dinner fork, small knife,* and *soup spoon.* (Recognition is performed within the low-level image processing unit.) In order to test whether the observed image conforms to the specification given earlier, we need to determine whether the spatial relationships *right of, left of, above,* and *below* hold between certain pairs of items. These relationships can be expressed very naturally in a declarative language, such as Prolog, but far less easily in a conventional language, for example, C.

Spatial relationships like those given earlier that are expressed in English can be translated automatically (by a Prolog program) into a Prolog program. The latter is then used when we examine the image, hoping to verify that the desired arrangement of cutlery and flatware is present.

This is a typical example of the kind of processing that is included under the term *high-level vision,* since it uses abstract (spatial) relationships between entities that are represented symbolically (e.g., *dinner fork, small knife, soup spoon,* etc.) Notice that we are effectively using English to program the vision system. This has obvious attractions compared with a conventional programming language, such as C. Using a computer to understand unconstrained natural language is still impossible but it is a straightforward matter to write a program that analyzes sentences relating to a subject of limited scope, such as operating a robot that moves pieces around a chessboard.

User and Machine Interfaces. Machine vision systems are usually intended to operate nearly autonomously but they never work in complete isolation from human beings or other machines. The user–machine interface is of particular importance and can make a crucial difference to the performance of a system. There are several ways in which a person needs to interact with an inspection system:

1. A vision system may be programmed on the factory floor to accommodate new products or new situations. Knowledge about the nature of a "good" product and/or typical defects must somehow be incorporated into a program to control the vision system. It should not be taken for granted that a ubiquitous language, such as C, is best suited for this, since knowledge of this kind is often expressed in abstract symbolic terms. Careful attention be paid to the user interface and, as far as possible, in situ reprogramming should not be expected to depend on shop-floor personnel using formal computer languages. Visual programming, combined with a touch-sensitive screen, mouse, joy pad, light pen, etc., together with menus, iconic displays, custom-designed dialog boxes,

etc., are much to be preferred. Speech input and output offers an attractive alternative.

2. Natural language input, using declarative programming techniques, offers a natural way for a human being to program a vision system. The mental effort involved is similar to that needed when describing an object that has been lost to a friend, so that he or she can help to find it; it is sufficient to describe the object and allow the vision system (i.e., the friend or the machine) to hunt for it without further instruction.
3. Performance statistics are important for monitoring the overall health of a manufacturing plant. The vision system should be able to display its recent performance statistics to the shop-floor personnel in an easily assimilated format. This should preferably use graphical or pictorial display formats, rather than tables of numbers. It should be possible, for example, to find what *reject* and *accept* rates the vision system has found during the last hour or day and to compare this with the results of previous days.
4. Factory-floor personnel will trust a machine more if they "understand" it and are much more likely to *feel* that they do so if they can examine the result of each step in a sequence of image-processing operations. An image-processing algorithm for machine vision is likely to be expressed in the form of a sequence of fairly simple steps. It requires very little effort or equipment to display the results of each of these steps, if requested to do so, by a user. This is helpful because people are very good at recognizing equipment and program malfunction if they are able to examine a series of pictures and compare each one with what their experience has taught them to expect for normal operation. Being able to display intermediate images, as well as the final results, gives the shop-floor personnel a much greater degree of confidence than they would otherwise have. To achieve good ergonomics, the machine should be able to freeze pictures for as long as the user wishes and to perform single-step operation, so that the user can work at his or her own pace.

To summarize, vision systems should be multimedia devices rather than simple image processors. This is true for both robot-vision and -inspection machines.

A vision system must also cooperate with a variety of external devices, such as multiaxis robots, (x, y, θ) tables, solenoids, valves, relays, simple *accept* or *reject* mechanisms, lamps, cameras, and lense. It is important therefore that the system has an appropriate interface. Serial (RS 232/RS 423), parallel, SCSI, IEEE 488, Ethernet, and Personal Computer Memory Card International Association (PCMCIA) are the most useful in practice. Good engineering practices should be adopted with opto-isolated interfaces used where possible for safety.

Other Aspects of Systems Engineering. Eliminating noise in all of its various forms is essential if we are to achieve consistent and reliable performance from a vision system. Dirt is one of the principal causes of malfunction. For this reason, it is imperative that great care be taken to keep lamps, all optical surfaces (including the light source and its reflector or diffuser), and the camera clean. Figure 4 shows how the camera and its associated optics can be protected from airborne dust, fumes, hot air, and infrared radiation.

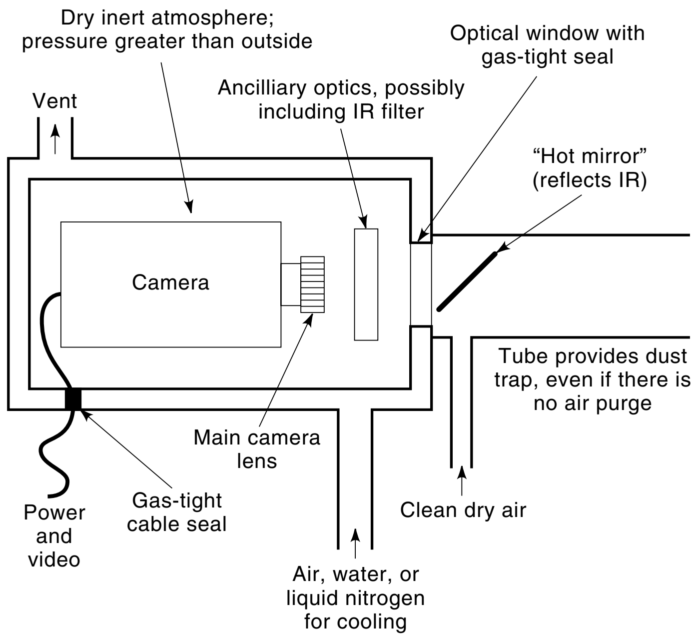

Figure 4. Protecting a camera and its associated optics. In practice, a selection of these measures is usually sufficient to provide adequate protection.

Fiber optics can be useful to keep the light source away from the point of inspection. This has the additional benefit of reducing the level of infrared radiation in the applied lighting, thereby protecting heat-sensitive materials. Similarly, fiber-optic imaging devices (intrascopes and fiberscopes) allow the camera to be removed from danger. Of course, there are various other ways to do this, most obviously using mirrors and long-focal-length lenses, to place the camera out of harm's way.

The worst effects of stray light can be eliminated by fitting suitable screens around the camera and scene to be viewed. This is usually a far cheaper alternative than increasing the sophistication of the image-processing hardware or software to accommodate variable lighting. Unfortunately, not all equipment suppliers, or their customers, follow this maxim.

The light source should be driven from a stabilized, well-regulated power supply. It is advisable to use a photodetector to monitor the light output from the lights and use a closed-loop control system to maintain the light level at a constant level, throughout the working life of the lamps. (Power–light conversion efficiency varies enormously during the lifetime of most types of lamp.)

Electrical cables should be screened and earth loops eliminated to avoid inducing noise into electronic circuits. This is especially important for the analog (video) signal lines. Wherever possible, digital signal lines should be opto-isolated for safety.

Concluding Remarks

Machine-vision systems are being used in manufacturing industry in a very diverse range of situations. To date, the electronics industry has been the most enthusiastic user of machine vision, with strong support coming from other industries, including automobile, glass, plastics, aircraft, printing, pharmaceutical, food, domestic and office goods, and medical products. The technology has already shown itself to be capable of contributing to a very wide range of applications, but there are problems caused by the large amount of skilled engineering effort needed to tailor a machine to suit each particular application. Improved design aids are needed to circumvent this bottleneck, and these are currently being developed. This bottleneck currently provides one of the main obstacles to further growth in the technology.

There is a never-ending quest for ever higher computing speeds in the image-processing subsystem, and some of the modern developments make the future look very promising. Field programmable gate arrays (FPGAs), reduced instruction set computing (RISC), pipelined image-processing hardware, custom integrated circuits, and algorithmic developments have all made vision a much more attractive technology in recent years. However, one of the most important developments of all that has taken place in recent years is simply confidence; each successful application seems to spawn at least two others. Vision equipment supply companies are no longer responding to every inquiry from a would-be customer as if their lives depend upon the outcome. Equipment vendors are far more confident in themselves and instill greater confidence in their customers than they did even five years ago. Despite this optimism, there remains an acute skill shortage in the machine-vision industry. We could solve many more problems if we had more people to think about them. The need for improved design aids is acute.

IMAGE PROCESSING FOR MACHINE VISION

Representations of Images

We shall first consider the representation of *monochrome* (gray-scale) images. Let i and j denote two integers where $1 \le i \le m$ and $1 \le j \le n$. In addition, let $f(i, j)$ denote an integer function such that $0 \le f(i, j) \le W$. (W denotes the white level in a gray-scale image.) An array F will be called a *digital image,* if

$$F = \begin{vmatrix} f(1,1) & f(1,2) & f(1,n) \\ f(2,1) & f(2,2) & f(2,n) \\ f(m,1) & f(m,2) & f(m,n) \end{vmatrix}$$

An address (i, j) defines a position in F, called a *pixel, pel,* or *picture element.* The array F contains a total of mn elements and this product is called the *spatial resolution* of F. We may *arbitrarily* assign intensities according to the scheme in Fig. 5.

A gray-scale image requires storing $\lceil \log_2(1 + W) \rceil$ bits. (Notice that $\lceil \ldots \rceil$ denotes the ceiling function.) A *binary image* is one in which only two intensity levels, black (0) and white (1), are permitted. This requires the storage of mn bits/image. An impression of color can be conveyed to the eye by superimposing *three* separate images and requires $3mn\lceil \log_2(1 + W) \rceil$ bits. A color video sequence that displays p frames per second requires $3mnp\lceil \log_2(1 + W) \rceil$ bits/s.

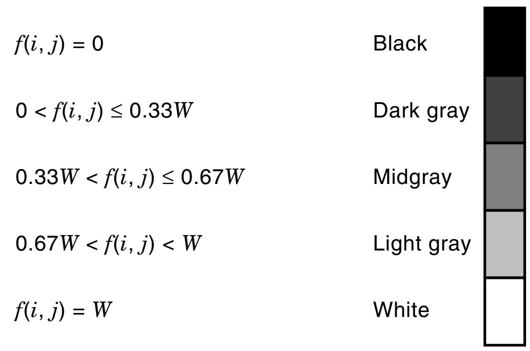

Figure 5. Scheme for assigning intensities.

Low-Level Image Processing Operators (Gray-Scale Images)

The following notation will be used throughout this section, in which we shall concentrate upon gray-scale images, unless otherwise stated.

1. $A = \{a(i, j)\}$, $B = \{b(i, j)\}$, and $C = \{c(i, j)\}$. A and B are input images. C is an output image.
2. W denotes the white level. (In a color image, W denotes the maximum output value on each of the color separation channels.)
3. $g(X)$ is a function of a single independent variable X.
4. $h(X, Y)$ is a function of two independent variables X and Y.
5. The assignment operator $\Leftarrow$ will be used to define an operation that is performed upon all pixels within an image.
6. k_1 and k_2 are constants.
7. $N(i, j)$ is that set of pixels arranged around the pixel (i, j) in the following way:

$(i-1, j-1)$	$(i-1, j)$	$(i-1, j+1)$
$(i, j-1)$	(i, j)	$(i, j+1)$
$(i+1, j-1)$	$(i+1, j)$	$(i+1, j+1)$

   Notice that $N(i, j)$ forms a 3×3 set of pixels and hence is referred to as the *3×3 neighborhood* of (i, j). Other neighborhoods can be defined but, in this introduction to the basics of the subject, we shall limit ourselves to consider only 3×3 windows. The points $\{(i-1, j-1)$, $(i-1, j)$, $(i-1, j+1)$, $(i, j-i)$, $(i, j+1)$, $(i+1, j-1)$, $(i+1, j)$, $(i+1, j+1)\}$ are called the *8-neighbors* of (i, j) and are also said to be *8-connected* to (i, j). The points $\{(i-1, j)$, $(i, j-1)$, $(i, j+1)$, $(i+1, j)$ are called the *4-neighbors* of (i, j) and are said to be *4-connected* to (i, j).
8. $\#(i, j)$ denotes the number of white points within the set $N(i, j)$, including (i, j) itself.

Monadic, Point-by-Point Operators. These operators have a characteristic equation of the form

$$c(i, j) \Leftarrow g(a(i, j))$$

There is one input image, $A = \{a(i, j)\}$, while the output image is $C = \{c(i, j)\}$. It is important to realize that $c(i, j)$ depends upon only $a(i, j)$. Examples of this type of operator include negate, threshold, square, square root, logarithm, and exponent. They can all be implemented using a look-up table (software), random access memory (RAM), or read only memory (ROM) (hardware).

Dyadic, Point-by-Point Operators. Dyadic operators have a characteristic equation of the form

$$c(i, j) \Leftarrow h(a(i, j), b(i, j))$$

There are two input images, $A = \{a(i, j)\}$ and $B = \{b(i, j)\}$, while the output image is $C = \{c(i, j)\}$. Notice that $c(i, j)$ depends upon only $a(i, j)$ and $b(i, j)$. The most important dyadic operators are addition, subtraction, multiplication, maximum, and minimum. The dyadic operators can be implemented in a standard digital arithmetic and logic unit (ALU) or a look-up table.

Linear Local Operators. The following equation defines a local operator which uses a *3 × 3 processing window:*

$$\begin{aligned} c(i, j) \Leftarrow [&W_1 a(i-1, j-1) + W_2 a(i-1, j) + W_3 a(i-1, j+1) \\ &+ W_4 a(i, j-1) + W_5 a(i, j) + W_6 a(i, j+1) \\ &+ W_7 a(i+1, j-1) + W_8 a(i+1, j) \\ &+ W_9 a(i+1, j+1)]k_1 + k_2 \end{aligned}$$

$W_1, W_2, \ldots, W_9$ are weights, which may be positive, negative, or zero. Values for the constants k_1 and k_2 are chosen to ensure that the range of the output $\{c(i, j)\}$ is the same as that of the inputs $\{a(i, j)\}$. This process is termed *normalization* and is an essential feature in all image-processing systems, to avoid overflow and underflow. (Normalization is also applied to other classes of image-processing operators.)

The following rules summarize the behavior of this type of operator:

1. If all weights are either positive or zero, the operator will *blur* the input image. Blurring is referred to as *low-pass filtering*. The larger the processing window, the greater will be the blurring effect.
2. Subtracting a blurred image from the original results in highlighting those small spots at which the intensity is markedly different from the background (local intensity anomalies. This is termed *high-pass filtering*.
3. If $W_1 = W_2 = W_3 = W_7 = W_8 = W_9 = 0$ and W_4, W_5, $W_6 > 0$, then the operator blurs along the rows of the image; horizontal features, such as edges and streaks, are not affected. It is of course possible to blur along the columns of the image, in which case the vertical features are not affected.
4. If $W_2 = W_3 = W_4 = W_6 = W_7 = W_8 = 0$ and W_1, W_5, $W_9 > 0$, then the operator blurs along the diagonal (top left to bottom right). There is no smearing along the orthogonal diagonal.
5. Repeating a low-pass operator, thereby increasing the blurring effect, may be represented instead by a local operator that uses a larger processing window. Repeating a blurring operator defined in a $(2n + 1) \times (2n + 1)$ window a total of R times is exactly equivalent to blurring using a certain operator defined within a $(2nR + 1) \times (2nR + 1)$ window.

Nonlinear Local Operators

Largest Intensity Neighborhood Function. The following operator has the effect of spreading bright regions and contracting dark ones:

$$\begin{aligned} c(i, j) \Leftarrow \mathbf{MAX}(&a(i-1, j-1), a(i-1, j), \\ &a(i-1, j+1), a(i, j-1), a(i, j), a(i, j+1), \\ &a(i+1, j-1), a(i+1, j), a(i+1, j+1)) \end{aligned}$$

Edge Detector. This operator highlights edges (i.e., points at which the intensity is changing rapidly). Notice how similar it is to the largest intensity neighborhood function,

$$\begin{aligned} c(i, j) \Leftarrow \mathbf{MAX}(&a(i-1, j-1), a(i-1, j), \\ &a(i-1, j+1), a(i, j-1), a(i, j), a(i, j+1), \\ &a(i+1, j-1), a(i+1, j), a(i+1, j+1)) - a(i, j) \end{aligned}$$

Median Filter. This filter is particularly useful for reducing the level of specklelike noise in an image.

$$\begin{aligned} c(i, j) \Leftarrow \mathbf{FIFTH\_LARGEST}(&a(i-1, j-1), \\ &a(i-1, j), a(i-1, j+1), a(i, j-1), a(i, j), \\ &a(i, j+1), a(i+1, j-1), a(i+1, j), a(i+1, j+1)) \end{aligned}$$

Sobel Edge Detector. This popular operator highlights the edges in an image; points where the intensity gradient is high are indicated by bright pixels in the output image:

$$\begin{aligned} c(i, j) \Leftarrow \{&|a(i-1, j-1) + 2a(i-1, j) + a(i-1, j+1) \\ &- a(i+1, j-1) - 2a(i+1, j) - a(i+1, j+1)| \\ &+ |a(i-1, j-1) + 2a(i, j-1) + a(i+1, j-1) \\ &- a(i-1, j+1) - 2a(i, j+1) - a(i+1, j+1)|\}/6 \end{aligned}$$

***N*-Tuple Operators.** *N*-tuple operators may be regarded as generalized versions of local operators. Let us consider a *linear* local operator that uses a large processing window (say, *rs* pixels) with most of its weights equal to zero. Only N of the weights are nonzero, where $N \ll rs$. This is an *N*-tuple filter. The *N*-tuple filters are usually designed to detect specific patterns. In this role, they are able to locate a simple feature, such as a corner, annulus, the numeral 2, etc., in any position. Notice how the goodness of fit varies with the shift, tilt, size, and font. Another character (Z in this case) may give a score that is close to that obtained from a 2, thus making these two characters difficult to distinguish reliably (Fig. 6).

Nonlinear tuple operators may be defined in a fairly obvious way. For example, we may define operators that compute the average, maximum, minimum, or median values of the intensities of the N pixels covered by the *N*-tuple. An important class of such functions is the *morphological operators,* including dilate and erode, described below.

Edge Effects. All local operators and *N*-tuple filters are susceptible to producing peculiar effects around the edges of an image. The reason is that to calculate the intensity of a

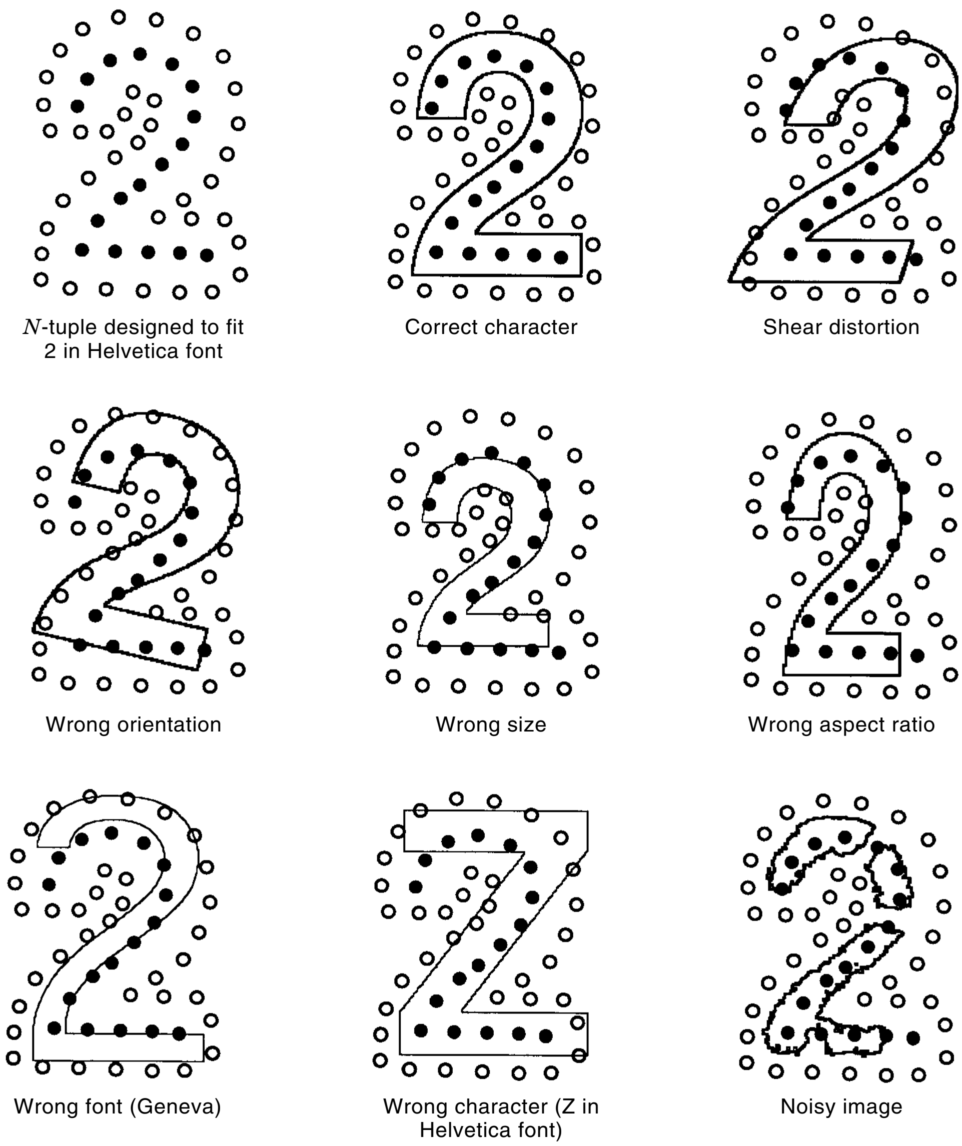

Figure 6. Recognizing a numeral 2 using an N-tuple operator. Notice how the goodness of fit varies with scale, aspect ratio, shape warping, orientation, font, and image noise.

point near the edge of an image, we require information about pixels outside the image, which of course are simply not present. In order to make some attempt at calculating values for the edge pixels, it is necessary to make some assumptions, for example, that all points outside the image are black, white, or have the same values as the border pixels. Whatever strategy we adopt is perfectly arbitrary, and there will be occasions when the edge effects are so pronounced that there is nothing that we can do but to remove them by masking. Edge effects are important because we have to make special provisions for them when we try to "patch" filtered images together.

Intensity Histogram. The *intensity histogram* of an image is the graph of the frequency of occurrence of each intensity value (i.e., the number of pixels with that intensity) in that image plotted against the intensity. A related function, called the *cumulative histogram,* can be calculated by integrating the intensity histogram. Let $h(p)$ denote the intensity histogram of a given image, where p is the intensity. Then, the cumulative histogram $H(p)$ is found by applying the following relation recursively:

$$H(p) = H(p-1) + h(p), \qquad H(0) = h(0)$$

Both $h(p)$ and $H(p)$ have a great many uses, since they summarize the statistics of the image. One of the main uses of $h(p)$ is in selecting a parameter for a threshold operator. It is often found, for example, that a suitable value for the threshold parameter can be related directly to the position of the "foot of a cliff" or to a "valley" in the graph of $h(p)$ against p. In addition, it is possible to determine that intensity level, which, when used as a threshold parameter, ensures that a given proportion of the output image is black. Thus, the mean, quartile, and decile intensities can be derived easily from $H(p)$, as can the standard deviation of the intensities.

Histogram equalization is an important process for enhancing the contrast of an image. It is based on an intensity-mapping function derived by a simple linear rescaling of $H(p)$. The intensity histogram of the image derived by applying the mapping function $H(p)/mn$ to the image from which $H(p)$ was derived is very nearly flat. (The resolution of the input image is mn pixels.) An image that has been transformed by histogram equalization can often show a great deal more detail to the eye than was evident in the original image.

The operator known as the *local area histogram equalization* is a method of filtering an image that is especially good at enhancing textures, at the expense of obscuring sharp intensity transitions (edges). This technique depends upon the application of histogram equalization within a processing window of moderate size, typically 5 × 5 to 25 × 25. The number of pixels in that window that are darker than the central pixel is found and defines the intensity at the equivalent point in the output image. The processing window is scanned across the image, just as it is for a local or n-tuple operator, and the calculation just described is repeated for each pixel.

Low-Level Image-Processing Operators (Binary Images)

Despite their apparent simplicity, compared with gray-scale pictures, the processing or analysis of binary images is far from trivial because there are innumerable types of features that we would like to identify and measure. For example, when designing a machine to recognize printed numerals, we must detect and locate features such as corners, limb ends, T junctions, loops, and enclosed areas. When we use a vision system to analyze the silhouettes of bottles and jars, we need to identify the straight vertical sides, shoulder, neck, mouth, and base. To summarize, binary image-processing operators effectively provide the link between the low-and the high-level phases of image understanding.

Binary Images. In this section, it will be convenient to assume that $a(i, j)$ and $b(i, j)$ can have only two values: 0 (black) and 1 (white). The operator + denotes the Boolean OR operation, $\cdot$ represents the AND operation, and $\oplus$ denotes the Boolean exclusive-OR operation.

Inverse (Binary Image Negation)

$$c(i, j) \Leftarrow \mathbf{NOT}[a(i, j)]$$

AND

$$c(i, j) \Leftarrow a(i, j) \cdot b(i, j)$$

OR

$$c(i, j)\, a(i, j) + b(i, j)$$

Exclusive OR

$$c(i, j) \Leftarrow a(i, j) \oplus b(i, j)$$

Expand White Areas (Dilate)

$$\begin{aligned} c(i, j) \Leftarrow\ & a(i-1, j-1) + a(i-1, j) + a(i-1, j+1) \\ & + a(i, j-1) + a(i, j) + a(i, j+1) + a(i+1, j-1) \\ & + a(i+1, j) + a(i+1, j+1) \end{aligned}$$

Shrink White Areas (Erode)

$$\begin{aligned} c(i, j) \Leftarrow\ & a(i-1, j-1) \cdot a(i-1, j) \cdot a(i-1, j+1) \cdot a(i, j-1) \\ & \cdot a(i, j) \cdot a(i, j+1) \cdot a(i+1, j-1) \\ & \cdot a(i+1, j) \cdot a(i+1, j+1) \end{aligned}$$

Edge Detector

$$\begin{aligned} c(i, j) \Leftarrow a(i, j) \cdot \mathbf{NOT}[& a(i-1, j-1) \cdot a(i-1, j) \\ & \cdot a(i-1, j+1) \\ & \cdot a(i, j-1) \cdot a(i, j) \cdot a(i, j+1) \\ & \cdot a(i+1, j-1) \\ & \cdot a(i+1, j) \cdot a(i+1, j+1)] \end{aligned}$$

Remove Isolated White Points

$$c(i, j) \Leftarrow \begin{cases} 1 & a(i, j).[\#(i, j) > 1] \\ 0 & \text{otherwise} \end{cases}$$

Count White Neighbors

$$c(i, j) \Leftarrow \#[a(i, j) = 1]$$

In this instance, $\#(Z)$ is the number of times that the logical variable Z is true. Notice that $\{c(i, j)\}$ is a gray-scale image.

Connectivity Detector. Consider the following pattern:

1	0	1
1	X	1
1	0	1

If $X = 1$, then all of the 1's are 8-connected. Alternatively, if $X = 0$, then they are not connected. In this sense, the point marked X is critical for connectivity and is therefore assigned the value 1 in the output image. This is also the case in the following examples:

1	0	0
0	X	1
0	0	0

1	1	0
0	X	0
0	0	1

0	0	1
1	X	0
1	0	1

However, those points marked X below are not critical for connectivity, since setting $X = 0$ rather than 1 has no effect on the 8-connectivity of the 1's.

1	1	1
1	X	1
0	0	1

0	1	1
1	X	0
1	1	1

0	1	1
1	X	0
0	1	1

A connectivity detector shades the output image with 1's to indicate the position of those points that are critical for connectivity and that were white in the input image. Black points and those that are not critical for connectivity, are mapped to black in the output image.

Euler Number. The Euler number is equal to the number of connected components (blobs) minus the number of holes in a binary image. Let us define three numbers N_1, N_2, and N_3, where N_i indicates the number of times that one of the patterns in the following pattern sets ($i = 1, 2$, or 3) occur in the input image.

Pattern Set 1 (N_1)

0	0
0	1

0	0
1	0

1	0
0	0

0	1
0	0

Pattern Set 2 (N_2)

0	1
1	0

1	0
0	1

Pattern Set 3 (N_3)

1	1
1	0

1	1
0	1

0	1
1	1

1	0
1	1

The *8-connected* Euler number, in which holes and blobs are defined in terms of 8-connected figures, is defined as: $(N_1 - 2N_2 - N_3)/4$. It is possible to calculate the *4-connected* Euler number using a slightly different formula: $(N_1 - 2N_2 + N_3)/4$. However, this parameter can give results that seem to be anomalous when we compare them to the number of holes and blobs counted by a human being.

Filling Holes. Consider a white bloblike figure containing a hole (lake) against a black background. The application of the hole-filling operator will cause all of the holes to be *filled in* by setting all pixels in the holes to white.

Region Labeling. Consider an image containing a number of separate bloblike figures. A region-labeling operator will shade the output image so that each blob is given a separate intensity value. We could shade the blobs according to the order in which they are found, during a conventional raster scan of the input image. Alternatively, the blobs could be shaded according to their areas; the biggest blobs becoming the brightest.

Other Methods of Detecting or Removing Small Spots. A binary image can be represented by a gray-scale image in which there are only two gray levels, 0 and W. Applying a conventional low-pass (blurring) filter to it results in a gray-scale image in which there is a larger number of possible intensity values. Pixels that were well inside large white areas in the input image are mapped to very bright pixels in the output image. Pixels that were well inside black areas are mapped to very dark pixels in the output image. However, pixels that were on the edge of a large blob or were inside small white spots in the input image are mapped to midgray intensity levels. Hence, thresholding can be used to eliminate the smaller spots, while at the same time, achieving a certain amount of edge smoothing of the large bright blobs, which remain.

Grass-Fire Transform. Consider a binary image containing a single white blob. Imagine that a fire is lit at all points around the blob's outer edge and the edges of any holes it may contain. The fire will burn inwards, until at some instant, advancing fire lines meet. When this occurs, the fire becomes extinguished locally. An output image is generated and is shaded in proportion to the time it takes for the fire to reach each point. Background pixels are mapped to black.

Skeleton. Consider a single white blob and a "bug" that walks around the blob's outer edge, removing one pixel at a time. No edge pixel is removed, if by doing so we would break the blob into two disconnected parts. In addition, no white pixel is removed, if there is only one white pixel among its 8-neighbors. This simple procedure leads to an undesirable effect in those instances when the input blob has holes in it; the skeleton that it produces has small loops in it that fit around the holes like a tightened noose. More sophisticated algorithms have been devised that avoid this problem.

Edge Smoothing and Corner Detection. Many methods of edge smoothing are possible. For example, we may map white pixels that have fewer than, say, three white 8-neighbors to black. This has the effect of eliminating "hair" around the edge of bloblike figure. Another technique described previously for eliminating small spots offers another possibility. A third option is to use the processing sequence: *expand, shrink, shrink, expand,* where *expand* represents expansion of white areas and *shrink* denotes shrinking of white areas. An algorithm based on following the edges provides another possibility and is fast in operation.

Convex Hull. Consider a single blob in a binary image. The convex hull is that area enclosed within the smallest convex polygon that will enclose the shape. This can be envisaged as the region enclosed within an elastic string, stretched around the blob. The area enclosed by the convex hull but not within the original blob is called the *convex deficiency,* which may consist of a number of disconnected parts and includes any holes and indentations. If we regard the blob as being like an *island,* we can understand the logic of referring to the former as *lakes* and the latter as *bays.*

Measurements on Binary Images. The following are just a few of the numerous descriptors that have been proposed for representing shapes in a binary image.

- Area.
- Perimeter.
- Distance of the furthest point on the edge of the blob from the centroid.
- Distance of the closest point on the edge of the blob from the centroid.
- Number of protuberances, as defined by that circle whose radius is equal to the average of the two parameters just defined.
- Distance of points on the edge of the blob from the centroid, as a function of angular position. This describes the silhouette in terms of polar coordinates. (This is not a single-valued function.)
- Circularity = (area)/(perimeter)^2. This will tend to zero for irregular shapes with ragged boundaries and has a maximum value ($=1/4\pi$) for a circle.
- Number of holes.
- Number of bays.
- Euler number.

- Ratio of the areas of the original blob and that of its convex hull.
- Ratio of the areas of the original blob and that of its circumcircle.
- Ratio of the area of the blob to the square of the total limb length of its skeleton.
- Distances between joints and limb ends of the skeleton.
- Ratio of the projections onto the major and minor axes.

There are numerous other shape measurements described in the literature.

Global Image Transforms

An important class of image-processing operators is characterized by an equation of the form $B \Leftarrow f(A)$, where $A = \{a(i, j)\}$ and $B = \{b(p, q)\}$. Each element in the output picture B is calculated using all or, at least a large proportion of the pixels in A. The output image B may well look quite different from the input image A. Examples of this class of operators are lateral shift, rotation, warping, Cartesian to polar coordinate conversion, and Fourier and Hough transforms. Further examples are discussed below.

Integrate Intensities Along Image Rows. This function is rarely of great value when used on its own, but it can be used to good effect when it is combined with other operators. The operator is defined recursively in the following way:

$$b(i, j) \Leftarrow b(i, j-1) + a(i, j)/n$$

where

$b(0, 0) = 0$

Row Maximum. This function is related to that just defined and is often used to detect local intensity minima. It is defined thus

$$c(i, j) \Leftarrow \mathbf{MAX}[a(i, j), c(i, j-1)]$$

Geometric Transforms. Algorithms exist by which images can be shifted, rotated, magnified (zoom in and out), can undergo axis conversion, and can be warped in other ways. Note that certain operations, such as rotating a digital image, can cause some difficulties because pixels in the input image are not mapped on a one-to-one basis to pixels in the output image. This can cause edges that are smooth in the input image to appear stepped after rotation. To avoid this effect, interpolation may be used, but this has the unfortunate effect of blurring edges.

The utility of certain coordinate axis transformations is evident when we are confronted with the examination of circular objects or those displaying spirals, concentric areas, or streaks radiating from a fixed point. Inspecting such objects is often made very much easier, if we first convert from *Cartesian* to *polar* coordinates. Warping is also useful in a variety of situations. For example, it is possible to compensate for *barrel* or *pin-cushion* distortion using quite simple warping functions. Geometric distortions introduced by a wide-angle lens or trapezoidal distortion due to viewing a scene from an oblique angle can also be corrected in a similar way.

Hough Transform. The Hough transform provides a powerful and robust technique for detecting lines and collinear collections of disjoint spots in a binary image. We shall describe the simplest version of the Hough transform, which is intended to detect straight lines. Circles, ellipses, parabolas, and other curves of predefined shape can be detected by extensions of the basic procedure described later.

A straight line in the input image is defined by the equation $r = x \cos \varphi + y \sin \varphi$, where r and φ are two unknown parameters whose values are to be found. Clearly, if this line intersects the point (x_i, y_i), then $r = x_i \cos \varphi + y_i \sin \varphi$. This equation can be solved for many different values of (r, φ). So, each white point (x_i, y_i) in the input image may be associated with a *set* of (r, φ) values. Actually, this set of points forms a *sinusoidal curve* in the (r, φ) plane. [The latter is called the *Hough transform* (HT) image.] Since each point in the input image generates such a sinusoidal curve in HT, the whole of the input image generates a multitude of overlapping sinusoids in the HT image. In many instances, a large number of sinusoidal curves are found to converge on the same spot in the HT image. The (r,φ) coordinates of such a bright point indicates the slope φ and position r of a straight line that can be drawn through a large number of white spots in the input image. In other words, a bright spot in HT corresponds to a white line, or a collinear collection of white points, in the input image.

Two-Dimensional Discrete Fourier Transform. The two-dimensional discrete Fourier transform (DFT) allows spatial periodicities of the intensity function of a given image to be investigated. This often finds application in analyzing textures of spongelike materials (e.g. industrial foams, baked food products, cakes, and bread) and examining surfaces of knitted and woven fabrics, as well as inspecting machined metal surfaces produced by milling or lanishing.

The two-dimensional DFT of an $N \times N$ image $f(x, y)$ is defined as follows:

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \exp[-j2\pi\,(ux + vy)/N]$$

where $0 \le u, v \le N - 1$. The inverse transform of $F(u, v)$ is defined as

$$f(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) \exp[j2\pi\,(ux + vy)/N]$$

where $0 \le x, y \le N - 1$. Several computational algorithms have been developed to calculate the two-dimensional DFT.

The Fourier power, or amplitude, spectrum plays an important role in image processing. Since the Fourier transform of a real function produces a complex function, $F(u, v) = R(u, v) + jI(u, v)$, the frequency spectrum of the image is the magnitude function

$$F(u, v) = [R^2(u, v) + I^2(u, v)]^{1/2}$$

and the power spectrum (spectral density) is defined as $P(u, v) = |F(u, v)|^2$. Both of these functions can be displayed, processed, and analyzed as an intensity image.

BIBLIOGRAPHY

1. Directory of resources relating to machine vision: http://www.eeng.dcu.ie/~whelanp/resources/resources.html (A dynamic reference library, based on the World Wide Web.)

Reading List

B. G. Batchelor and P. F. Whelan, *Intelligent Vision Systems for Industry,* London: Springer, 1997. (Provides a more detailed exposition of the ideas outlined here.)

B. G. Batchelor and P. F. Whelan (eds.), *Industrial Vision Systems,* SPIE Milestone Series, Vol. MS97, Bellingham, WA: SPIE—The International Society for Optical Engineering. (A collection of key technical articles that have shaped the development of industrial machine vision systems.)

BRUCE G. BATCHELOR
University of Wales Cardiff
PAUL F. WHELAN
Dublin City University
RALF HACK
University of Wales Cardiff

MACHINING, LASER BEAM. See LASER BEAM MACHINING.