Chapter 16

Case Studies

This chapter describes two different commercial systems that use computer vision and pattern recognition techiques to solve real application problems. These application solutions let us see an entire system design that integrates different hardware and algorithms. Most, but not all, methods used have been treated in some previous section of this textbook. The first case studied is IBM's VeggieVision system for identification of produce at the supermarket checkout station. The second is an iris identification system for verification or recognition of a person's identity at an ATM machine or secure facility.

16.1 Veggie Vision : A System for Checking out Vegetables

Use of bar codes has greatly reduced the amount of human labor required in selling products at the supermarket. Handling produce, however, continues to be labor intensive. Sometimes produce items, such as potatoes or apples, are packaged and bar-coded in advance so that they can be handled in the same way as canned or boxed products. Many items, however, are loose so that the customer can choose individual pieces; for example, tomatoes or even green beans. Loose items may or may not be put in a plastic bag by the customer. In a typical store, such items are placed on a scale to determine the weight and possibly the checker has to identify the product and type its code into the register. This problem begs for an automated solution — why not have a camera in the scale to automatically identify the type of produce? Such a system would greatly simplify the job of the human checker and improve inventory control. A system called VeggieVision has in fact been developed at the IBM T. J. Watson Research Center. Laboratory testing has shown the system to be effective, and it is now under field test. An automated system has other advantages, such as more refined pricing by produce size or ripeness. The sections below give more details of the supermarket produce problem and the solution developed by IBM. We give special acknowledgement to Ruud Bolle and Jonathan Connell for providing documentation of VeggieVision. The reader can consult their publications cited in the references for more details than are given below.

Figure 16.1: Designer's sketch of final supermarket system. (Likely produce items are displayed in case human interaction is needed.) Courtesy of R. Bolle and J. Connell.

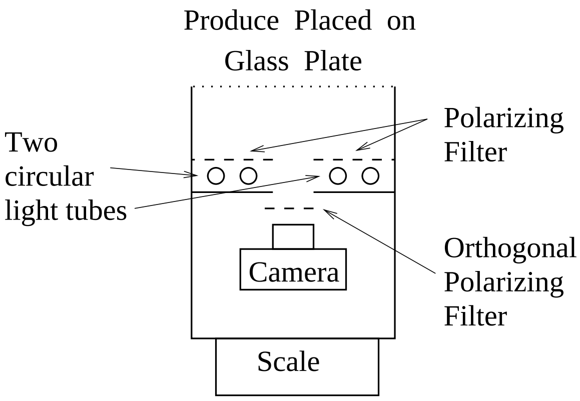

Figure 16.2: A vertical cross section of the scanner design shows color camera and polarizing light source within existing scale and bar-code reader.

16.1.1 Application Domain and Requirements

Roughly $m = 350$ different produce items are sold across the United States, but an individual market is likely to sell only about 150 items. Neither of these numbers present a difficult problem for automating produce identification. In order to be economical, an automated system must make an identification in about one second using computing equipment no more costly than what is already in current supermarket scanners and computers. It is desired that any new equipment be added into the same space as occupied by current store equipment and that no changes need to be made to the existing store environment.

The system must be adaptable to individual store environments because of several factors. First, not all stores carry the same produce items. Second, produce items change by season and even by day for the same store — consider, for example, a shipment of bananas that arrives somewhat green and gradually yellows. An effective system must be designed so that it can nicely adapt to such changes and have the capability of being extended to handle new items.

Finally, human operators must have an acceptable role in the operation of the overall system. This includes initial training to learn how to use the system, operation of the system in automatic mode, and making decisions when the automatic process is stymied for some reason. The overall system, including the machine and the human operator, must be much more efficient than the manual technique currently used in most stores. Figure 16.1 shows the desired overall system: a touch sensitive display is included to show results to the checker and to allow the checker to make an identification in case the automatic system is unsure.

16.1.2 System Design

Hardware Components

The scanning hardware had to fit within space similar to that already used for weighing produce and scanning bar-codes. Moreover, it had to operate in various store environments without any special alterations. Figure 16.2 shows the scanner that was designed. Polarizing filters were used on both the light source and the camera: the direction of the camera filter is orthogonal to the direction of the filter used on the illumination in order to remove specular reflections from the produce. A digital signal processing chip (DSP) was chosen to economically perform image processing operations within the one second cycle time target. The color camera and DSP act as a low rate input device for the cash register, which only needs an identifier and a weight for each item of produce placed on the scale.

Representation and Recognition

Previous applications showed that color histograms promised to be effective features: research and development confirmed this. Classical texture features did not perform well for this problem, so some problem-specific features were developed as described below. A simple shape feature was also used. Color, texture, shape and size histograms for a produce image were combined to form a feature vector Q of dimension $d > 100$ to represent

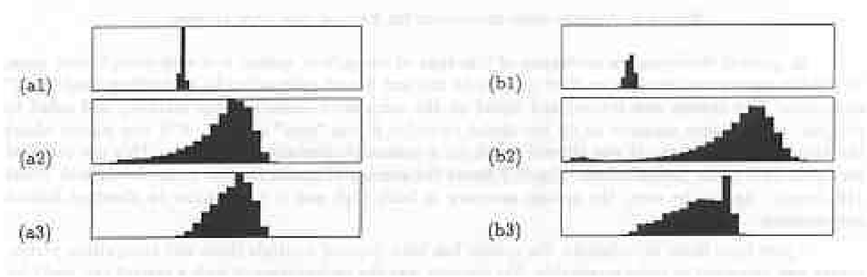

Figure 16.3: Color histograms for apples(left) and oranges(right). From top to bottom the histograms are for hue, saturation and intensity. Courtesy of R. Bolle and J. Connell.

the unknown produce on the scale. Figure 16.3 shows only the color histograms for some apples(left) and some oranges(right).

Due to the need for the system to adapt to varying conditions, a nearest neighbor classification scheme was chosen. Labeled samples of feature vectors from known produce items are stored in a simple memory array. With the maximum of $m = 350$ classes each with up to 10 samples, 3500 samples can be stored. The DSP can easily compare a query feature vector Q to all 3500 labeled samples within one second in order to find a set of the k nearest neighbors. There is no special data structure used to organize the training samples: this makes update simple. Each sample vector is stored with associated information so that the vector can be *aged* and possibly deleted from memory as the samples are used over time.

The distance between the query vector Q and the j-th training sample of class L, ${}^L P_j$ is computed as in Chapter 8. Since all the individual features are counts from a histogram, the distance $d(Q,^L P_j)$ is the absolute value of the different counts Q and ${}^L P_j$.

$$d^j \;=\; d(Q,^L P_j) \;=\; \sum_{f \in F} w_f \; d(Q,^L P_{j,f}) \tag{16.1}$$

Thresholds control the identification and determine its certainty. There is a distance threshold t that defines whether or not Q is *close enough* to some sample[s]P_j: let k_t be the number of neighbors close to Q selected from memory. The identification procedure is given in the next section.

16.1.3 Identification Procedure

The overall algorithm for identifying the produce on the scale is given in Algorithm 1. Some of the steps are described in more detail below. Use of the nearest neighbors in the training samples to identify the produce is sketched in Figure 16.4.

16.1.4 More Details on the Process

Obtaining Images of Produce

Identify the produce placed on the store checkout scale.

1. Upon operator signal, take lights-off image and lights-on image and extract foreground produce from background.
2. Make histogram of color features, texture features, shape features, and size features; concatenate them to form feature vector Q.
3. Compare Q to each training sample ${}^{L}P_j$ in memory; discard all differences larger than t; sort the remaining samples into ascending order.
4. If the closest K neighbors all have the same label L, then label L is returned as the identity of Q and the identification is *sure*. In this case, the system can make the decision automatically.
5. If the decision is not sure for the first time: ask the operator to reposition the produce and repeat the steps 1-4 above.
6. If the decision is not sure for the second time: display up to N choices of labels from the sorted list for operator decision.
7. If appropriate, contribute Q to the set of training samples and possibly delete other training samples.

Algorithm 1: Overall flow of produce identification used in VeggieVision

Images need to be taken with little control of the store environment. In particular, the camera within the scale will sense light from or reflecting from the ceiling. Two images are taken with the produce on the scale, one using the light source within the scale (lights-on) and one without this light source (lights-off). The three regions to be segmented are 1) the produce region is dark in the lights-off image and bright in the lights-on image, 2) the background region has similar brightness in both the lights-on and lights off images, and 3) if a plastic bag encloses the produce, it is not dark in the lights off image and not bright in the lights-on image. With this engineering, thresholds can be set so that the produce region is segmented out from the bag and background. To obtain quality color, polarized light is used within the scale to inhibit specular reflections, which will not be indicative of the produce surface. Moreover, the produce surfaces that are imaged do not receive uncontrolled illumination from outside the scale; thus, the sensed colors will be consistent even with variations of room lighting.

Computing Features

Features are computed only for the produce region[s]. Features must be rotation invariant, but not size invariant. Histograms of four types of features are created and concatenated to form a single vector Q representing the unknown produce. The four features are color, texture, shape, and size.

The **color** of each pixel is transformed from RGB representation to (h, s, i) in HSI. Each value h, s and i is contributed to a histogram: pixels where the intensity or saturation is low are not used due to numerical instability of the conversion. The three histograms are

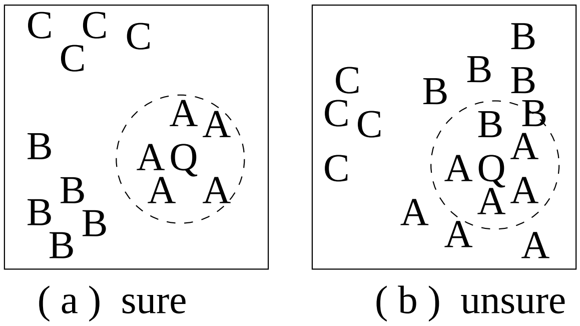

Figure 16.4: Sketch of concepts of decision-making in feature space: a 2D feature space is shown whereas the actual feature space is d-dimensional. (a) A "sure" identification results when all training samples within t of Q are from only one class. (b) The decision is unsure otherwise: either the produce is scanned anew, or the system asks the operator to choose from close classes A and B.

normalized with respect to the total area of the produce regions. Figure 16.3 shows the three separate color histograms for apples versus oranges.

A small set of **texture** features are computed at each pixel of only the green channel of the original color image. Center-surround masks of different sizes are used for computing texture features. Center-surround masks are just a center box region of positive weights and a surrounding background box region of negative weights. Computation is sped up by using a subsampled image. Both positive and negative responses to the masks are contributed to a histogram. The size of the central peak gives overall texture of the object; if the central peak is large, this means many low magnitude responses to the masks or very little texture. The spread of the histogram tells us how contrasty the texture is, e.g. sharp shadows between leaves versus subtle surface striping. Histogram asymmetry tells something about the size of the textons relative to the mask scale; for example, a skew toward positive differences suggests that the item has large leaves with narrow cracks between them versus something more "crinkly" such as parsley.

A simple scheme is used to measure **shape**. The boundaries of the produce regions are smoothed and followed to compute curvature at each boundary pixel. Only the external boundary segments are used; the image frame and places where produce items touch each other are not used. The square of each curvature value is contributed to a histogram in order to yield better clustering. Spherical produce should result in a narrow peak corresponding to the radius; the actual location can differentiate between lemons from grapefruits, for example. Elongated produce, such as bananas and carrots, result in a broader set of values and a peak near zero. Leafy vegetables give broad distributions of curvature.

The fourth feature histogrammed is **size**. A "size" value is computed for each foreground pixel and not just for aggregates of pixels. Run-lengths are computed within the binary foreground mask in four directions (horizontal, vertical, and the two diagonals), creating four directional images. In each directional image, the directional size of a pixel is the overall length of the run of which it is a part. The size of a foreground pixel is then taken to be the minimum directional value at that pixel. Object size is thus defined without parametric models and without any segmentation beyond the original foreground/background segmentation. A bunch of grapes segments into a "puffy cloud" foreground mask. Pixels on the outside bumps get small run-lengths while the interior pixels get much longer run-lengths. Thus the size histogram will have two peaks: one for the individual grapes, and one for the overall size of the bunch. A carrot will have a narrow peak in the size histogram at its characteristic width, which happens to be similar to the width of cherry tomatoes, and a peak around zero curvature in the shape histogram meaning that it is elongated, which tomatoes would not have.

Supervised Learning

The use of nearest neighbor classification provides for both bounded computation time and simple training and adaptation. Initially, the system can be trained by showing it several examples of the various produce items in the store and providing the class identification (inventory code). When the system is in use, the human operator can require that a new feature vector Q be added to the set of training samples. A sample can be deleted because it is redundant relative to the geometric structure or useage factor of the samples of its class. When training, a new sample that is correctly classified using existing samples is not memorized if it is within distance t_2 of the best match; otherwise, it is memorized. This allows multiple modes to form in feature space; for example, one for broccoli heads alone and one for broccoli with long stalks. When the limit of M samples has been reached for a class, the "least used" sample is erased. A useage count is incremented by $I+$ whenever a sample is the closest one and decremented by $I-$ when it is not.

Training VeggieVision from scratch in every store appears to be unnecessary. Experiments have shown that recognition performance will be depressed if training samples from a different store are used. However, the system can adapt, as described above. Human intervention will be high in the beginning as the system adjusts its training base to the produce actually leaving the store, but the overall human effort should be much less than beginning from scratch.

16.1.5 Performance

The developers have published the results of several experiments performed over a period of time. Details of the many variables studied can be found in the references. In a recent study using 5300 images over four different stores, the system was "sure" and correct 89% of the time, was either correct or had the correct category as the top choice for the operator 93% of the time, or was either correct or had the correct category listed somewhere in the first four operator choices 96% of the time. The operator may be asked once to reposition the produce. If VeggieVision is unsure in two tries, identification can be handled by the checker touching an icon on a display. It appears that the system would reduce labor considerably even if the checker would make all decisions using a touch-sensitive CRT.

Exercise 1

Assuming that bananas are rectangular, what should their shape histogram look like?

Exercise 2

Sketch and compare color, texture, and shape histograms expected for red apples versus yellow bananas.

Exercise 3

Suppose a customer combines 3 apples and 2 oranges in a plastic bag. Should our identification system be able to handle this? If so, how?

16.2 Identifying Humans via the Iris of an Eye

We now describe a system to identify persons by scanning the iris texture of an eye. The sensing hardware for an ATM environment is built by Sensar, which licenses from IriScan the software that performs feature extraction and matching. We give special aknowledgement to Gary Zhang of Sensar and John Daugman of Cambridge University for providing information and figures describing this application.

Identification of a person has always been an important problem for society. Correct identification must be established for commerical and legal transactions; for example, a person withdrawing cash from a bank account or changing an address of residence. Currently, this is often done by the person showing some document such as a driver's license or birth certificate to another person who controls in some way the action to be taken. In today's world many transactions are done via machines or computer networks: security and privacy are usually provided by the person (a) knowing a unique account number together with (b) a *password* or *PIN* associated with the account. These codes can be obtained, with or without permission, by other persons who may then do transactions thereby breaking the normal responsibility and control.

In addition to the very important applications in electronic commerce are those in police work. Fingerprints have been commonly used. Crime scenes are examined for fingerprints that might identify persons who were there. Fingerprints are also used in contexts where the person cooperates in the identification; for example, they are often used to identify workers in a secure environment. Fingerprints have been extensively studied and have used for over one hundred years. Several electronic devices have been developed so that fingerprints from a cooperating person can be easily input to computer networks or other systems (see Jain et al 1999). Face recognition is also under intense development for identification and authentication (authorization) systems. These evolving systems have the capability of identifying people without their knowledge, such as in an airport, bank, or hotel. Thus they have obvious use in police work and in providing security, but also are problematic in terms of acceptability and privacy.

16.2.1 Requirements for identification systems

We consider systems performing one of two kinds of operations: (a) to identify a person, cooperating or not, from a large set of possibilities, and (b) to confirm that a cooperating person is indeed the one he/she claims to be. The latter case is often called *authentication*. Some of the requirements are not obvious, so we include a list of them. The system design will be dominated by the particular *biometric* used and how it is sensed and encoded for machine use. Three important biometrics are the appearance of the human (1) fingerprint, (2) face, and (3) iris of the eye. It will be argued below that the iris of the eye provides better information than the fingerprint or face.

1. The system must obtain information from a human with minimal inconvenience.
2. The biometric code must have little variance as it is obtained from the same person over time.
3. The biometric code obtained from one person must be significantly different from that of other persons. (The set of persons to be discriminated will vary from one application to another.)
4. It must be very difficult to fool the system with "fake data", such as an image printed on paper.
5. The system must be cost effective relative to the particular application.

Before going on to describe the iris scanning system, it is instructive to make some comparisons among different biometrics regarding how they satisfy the above requirements. Besides those mentioned above, we add analysis of DNA as a biometric.

1. **convenience in obtaining information**: fingerprint(fair); face(good); iris(good); DNA(poor). Cheap digital scanners exist for fingerprints but the user must carefully present a finger to them. A face can be effectively sensed by a cheap video camera with little inconvenience: more expensive optics and more control is needed to obtain a quality image of the iris. Obtaining DNA is, of course, an expensive off-line lab procedure, usually reserved only for important court cases.
2. **low intraclass variance**: fingerprint(good); face(fair); iris(excellent); DNA(excellent). Note that strong deformations are added in taking fingerprints and that facial appearance can vary with pose, mood, hair and age. The iris develops its texture before a child is born and changes very little over life, and a scanning system has been developed to produce consistent encodings of it.
3. **high interclass variance**: fingerprint(good); face(good); iris(excellent); DNA(excellent). While fingerprints provide excellent discrimination in the hands of experts, they are not quite as good when represented for automatic methods. Most people have "doubles" in terms of facial appearance, especially twins. Twins can by themselves create a 1% error rate! And, twins have the same DNA. Interestingly, twins do not have the same iris texture. In fact, the textures of the two eyes *from the same person* are just as uncorrelated as are the textures from eyes of different individuals!
4. **difficult to fool**: fingerprint(good); face(good); iris(excellent); DNA(excellent). Several systems that use fingerprints or faces can be fooled by pictures or simple models of a person's appearance. The pupil of the eye, which is inside the iris, undergoes size changes that can be tracked by a sensing system as a check against subterfuge.
5. **cost effective**: fingerprint(fair);face(fair);iris(fair);DNA(poor). Fingerprints and faces can be sensed cheaply; however, methods of matching their representations can be expensive. Scanning the iris is expensive but matching is simple. Using DNA, of course, is expensive in time, human labor, and materials.

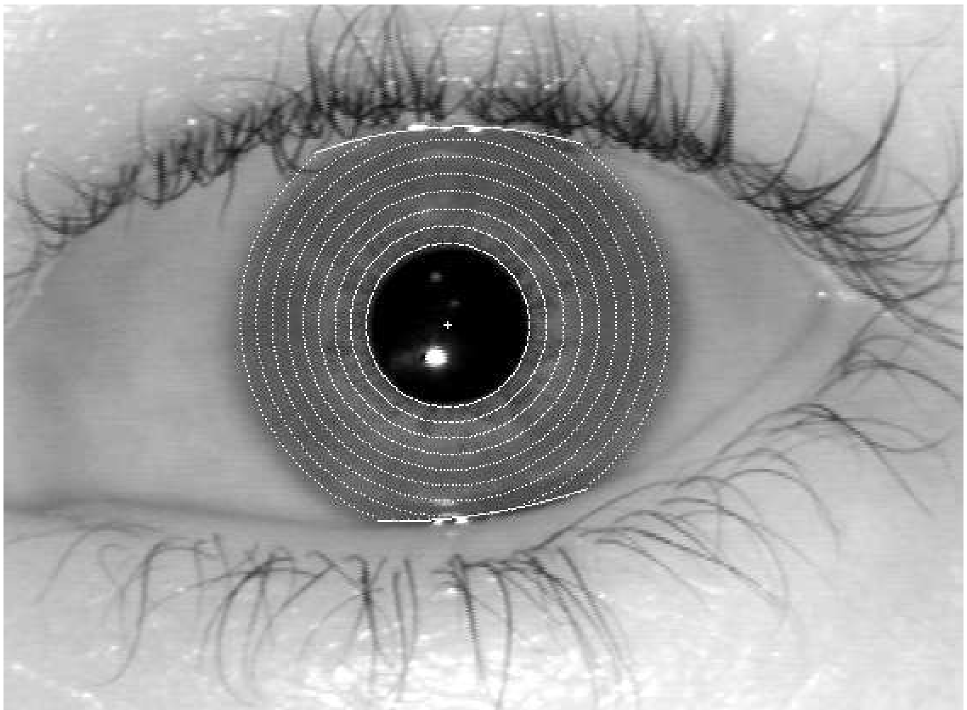

Figure 16.5: Narrow FOV image of the eye: image processing has identified 8 different bands of the iris from which texture features are extracted. (Contributed by John Daugman, Cambridge University.)

16.2.2 System Design

For concreteness, we describe the application of iris-scanning to identify users of an automatic teller machine (ATM). When the ATM user approaches the ATM machine, the iris of one eye is scanned and the *Sensar...Secure*^{RM} *System* identifies the person in the customer records. The customer may then have access to the account or perhaps is asked to type a password for additional security. Variations in the application, such as opening a secure door, can be handled by varying the design parameters set below.

Careful scanner design and special optics are needed in order to obtain a high resolution image of a such a small object relative to the large 3D FOV in which it might appear. Moreover, 3D analysis is required in order to find the front person in case there is a queue waiting to use the machine. Once the front person's eye is located and scanned, patented software is used to obtain a d = 2048-dimensional binary vector Q representing the grey-level texture of the iris. Matching this vector to a set of vectors representing some universe of persons is performed quite simply by computing minimum Hamming distance, which is just the number of bits in which two binary vectors differ.

Hardware Components

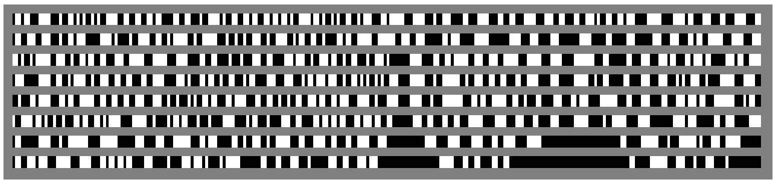

Figure 16.6: Graphical representation of 2048 bit code representing the signs of the results of applying Gabor filters of different sizes to locations within the 8 bands shown in Figure 16.5. (Contributed by John Daugman, Cambridge University.)

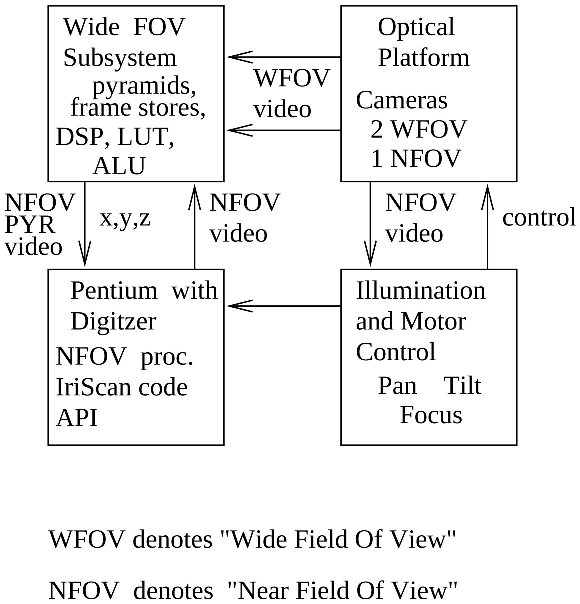

Figure 16.7: *Sensar...Secure*^{TM} distributed processing architecture.

A sketch of the *Sensar...Secure*^{TM} distributed processing architecture is given in Figure 16.7. The system is composed of four main units – 1) a general purpose computer that provides an interface with an application process and the sensing control and video processing units, 2) an optical platform with three cameras that obtains both wide field of view (WFOV) images and near field of view images (NFOV), 3) the control unit for the optical platform, and 4) the video processing unit, which has special hardware for real-time processing of stereo video.

The two wide field of view cameras obtain a video stream that is used to locate the frontmost person in the field of view. The two video streams are passed to the signal processing unit, which performs the actual stereo processing using multiresolution pyramids. The x-y-z location of the designated eye of the person is passed to the main unit, which then uses that information to control the optical platform so that it can obtain the near field of view imagery of the eye. This cycle can be performed every half second so that a slowly moving person can be tracked. The near field of view video is processed by the main unit so that the specific eye region can be located and the iris code extracted. The overall process is given in Algorithm 2.

Note that the sensing hardware is more complex than what is used in many other systems and this limits the applications. The high cost is primarily a result of the requirement that sensing is passive as far as the user is concerned: the user can move freely in the work envelope and the system must locate her. This requirement necessitates the wide field of view sensing to find the subject followed by the near field of view sensing to obtain the needed image of the eye. The need for real-time stereo implies special hardware, hence the two resolution pyramids to speed up correlation of points in the two video streams.

Representation

The ultimate representation of the eye and person is just a binary vector of dimension 2048 (256 bytes of storage). A graphical representation of one such vector is given in Figure 16.6: 0 is printed black and 1 is printed white. Each bit of the code is determined by the sign of the result of correlating a specific Gabor filter with a specific neighborhood of the iris image. It is thus very important that the eye image be normalized for rotation before correlation is performed.

As shown in Figure 16.8, each bit of the iris code is determined by correlating a 2D Gabor wavelet with the image of the iris at a particular location (ρ_0, ϕ_0) on the iris and with particular spread parameters α and β (demodulation). The cross section of the wavelet along the ρ direction is a Gaussian with spread α, while the cross section along direction ϕ is a Gaussian with spread β modulated by a sine wave. Each correlation of a wavelet with the picture function yields a complex number c as follows.

$$c = \int_\rho \int_\phi f(\rho, \phi) \; [\; e^{-j\,2\pi(\phi-\phi_0)} e^{-(\rho-\rho_0)^2/\alpha^2} e^{-(\phi-\phi_0)^2/\beta^2} \;] \; \rho \; d\rho \; d\phi \tag{16.2}$$

The complex correlation is converted to two bits of the iris code by testing its sign: $if(Re(c) \geq 0.0) \;\; then \;\; b_{real} = 1 \;\; else \;\; b_{real} = 0$ and $if(Im(c) \geq 0.0) \;\; then \;\; b_{img} = 1 \;\; else \;\; b_{img} = 0$. Clearly, any rotation of the image of the iris about the view direction will

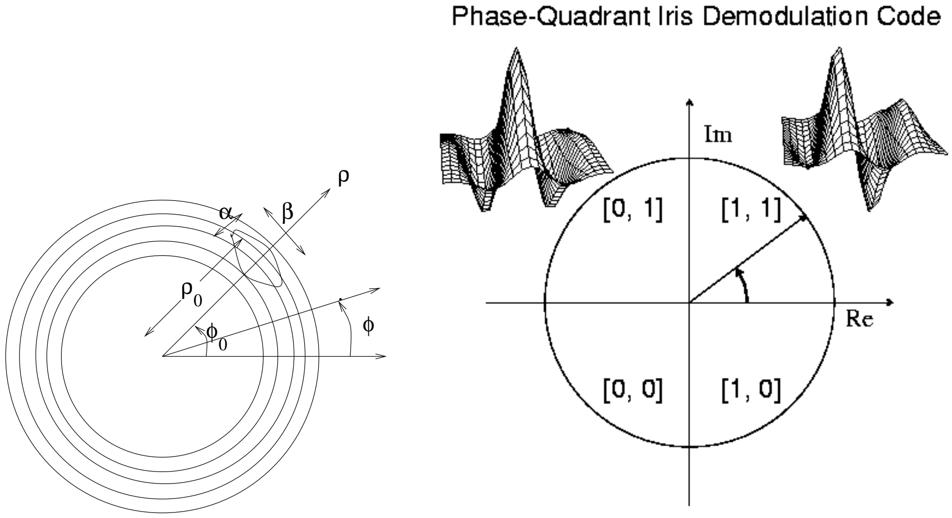

Figure 16.8: (Left) Sketch of annular regions of the iris defined by a radial range $\rho_a \leq \rho \leq \rho_b$ and placement of a Gabor wavelet at location (ϕ_0, ρ_0) with spread parameters α and β. (Right) Shape of complex valued 2D Gabor wavelet.(Contributed by John Daugman, Cambridge University.)

affect the location parameter ϕ_0. Any rotation will be small because the near field of view image can be obtained using information about the location of both eyes in the wide field of view image. During matching, the iris code is matched after undergoing slight rotations and the best match of these rotated codes to any code from the database of candidates is used. The ρ axis is defined in terms of the the boundary of the pupil and the outer boundary of the iris: these two boundaries are assumed to be circular but not necessarily with the same center. They are found by integrating edge evidence in much the same way that the circular Hough transform does. The boundaries are defined by the two sets of parameters ρ, x_c, y_c that maximize the gradient magnitude around the circle:

$$max_{(\rho, x_0, y_0)} \mid \frac{\partial}{\partial \rho} \oint \frac{f(x,y)}{2\pi\rho} ds \mid \tag{16.3}$$

Identification Procedure

A sketch of the identification procedure is given in Algorithm 2. Given our previous discussion, all important details of the procedure have been covered. Comments on performance are given in the next section.

16.2.3 Performance

The time for the system to acquire and identify an iris scan varies with conditions and normally is within the range of one to five seconds. This should satisfy the requirements for an ATM system, but may not be fast enough to process persons moving through current airport security systems, for example. For the ATM application, the mechanical aspects of camera

Compute identity ID of closest person of scene.

1. Using wide FOV video and correlation-based stereo, locate closest head.
2. Identify location of face features using templates; then identify left [or right] eye at $[x, y, z]$.
3. Using $[x, y, z]$, near FOV monochrome camera obtains in-focus centered image I of eye.
4. 2048 bit (256 byte) iris code Q obtained from image I of the eye using patented image processing software.
5. Iris code Q matched to those in database using XOR.
   If the two codes differ by fewer than K bits, then return person ID;
   else, return "reject".

Algorithm 2: Identification of a person enrolled in the database by iris scanning.

control are the limiting factors: perhaps 90% of the time is spent on acquiring the image. After the image is passed to the algorithms, it takes 200 msec to locate the iris boundaries and to generate the IrisCode. Matches proceed at the rate of about 100,000 persons/second.

Most important is the probabilty of the system making an error in identification. From a theoretical model fitted to many test cases, Daugman (1998) made the following estimates of error rates. If 70% of the 2048 bits must match in order to verify that the person is the one claimed, then the chances of accepting an imposter is about 1 chance in 6×10^9, while the chances of rejecting the true person is about 1 in 46,000. If the threshold is reduced to 66%, then the false accept and false reject rates are equal at about 1 chance in 1 million.

We conclude with a few words about the model used to estimate the above probabilties. The reader is referred to the Daugman paper for more detail. All possible pairs of iris scans from about 300 people were compared yielding the results shown in Figure 16.9. It was found that (a) when comparing different eyes, the distribution of Hamming distances (over 200,000 pairings) ranged between 0.4 and 0.6 of the bits, (b) a binomial distribution with N=266 degrees of freedom and $p = 0.5 = q$ fit the observed distribution very well, and (c) perhaps surprisingly, the same type of distribution was observed in comparing the left eye and right eye scans from the same people, showing that scans from the two eyes of the same person are as uncorrelated as the scans from two different people. Figure 16.9 plots the distribution of Hamming distances between codes of different persons (right mode) alongside the distribution of Hamming distances between codes from the same person (left mode) – the crossover point is at probability 10^{-6}. The decision threshold need not be set at the crossover point. If a distance of at most 30% of the bits is tolerated, then the probabilty of accepting an incorrect match is one in six billion, which might be desirable when the cost of a false accept is much greater than the cost of a false reject.

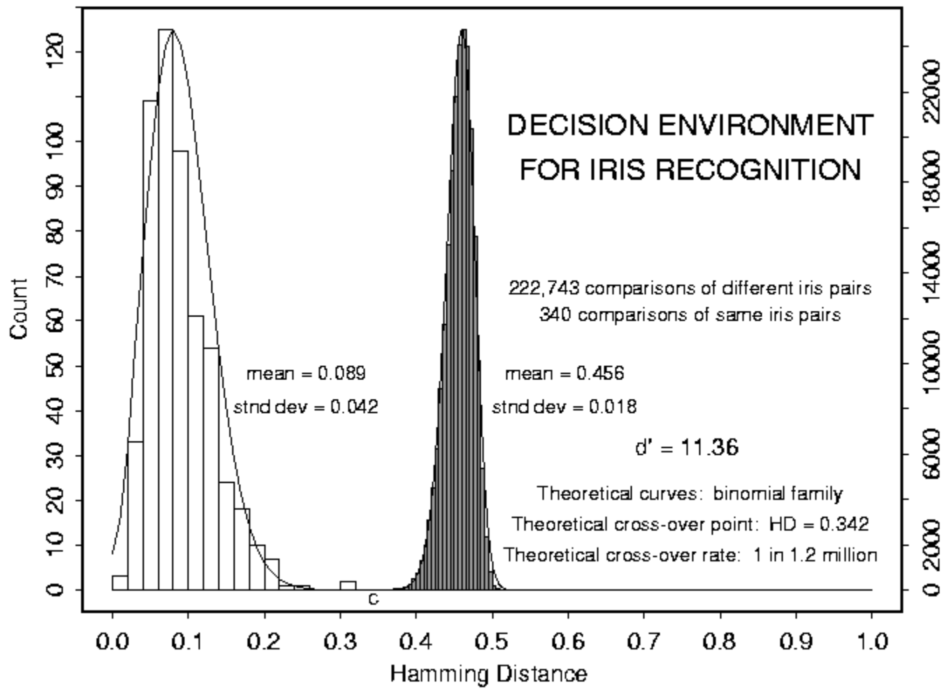

Figure 16.9: Distributions of Hamming distance for same eye (left mode) and different eyes (right mode). The crossover point is at 0.34 of the bits, where the probabilty of falsely dismissing a correct match equals the probability of falsely accepting an incorrect match: both are about 10^{-6}. (Figure contributed by John Daugman of Cambridge University.)

16.2.4 References

- R. Bolle, J. Connell, N. Haas, R.Mohan and G. Taubin (1996), *VeggieVision: A Produce Recognition System*, **Proc. IEEE Workshop on Applications of Computer Vision** 1996.
- J. Daugman (1998), *Recognizing Persons by Their Iris Patterns*, in **Biometrics: Personal Identification for a Networked Society**, A. Jain, R. Bolle and S. Pankanti (Eds), Kluwer Academic.
- J. Daugman (1994) "Biometric Personal Identification System Based on Iris Analysis." U.S. Patent No. 5,291,560 issued March 1, 1994 to John Daugman.
- M. DellaVecchia, T. Chmielewski, T. Camus, M. Salganicoff and M. Negin (1998), *Methodology and apparatus for using the human iris as a robust biometric*, in **Ophthalmic Technologies VIII: SPIE Proceedings**, (Jan 1998)24-30.