2000

A video human face tracker.

Yonghua (Walter). Jin
University of Windsor

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

A Video Human Face Tracker

By

Yonghua Jin (Walter)

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the Faculty of Engineering –
Electrical and Computer Engineering
in Partial Fulfillment
of the Requirements for the Degree of
Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

2000

Abstract

This thesis focuses on Automated Face Tracking, which is a major challenge for applications related to intelligent surveillance, law enforcement, telecommunications, human-computer interactions and virtual reality interfaces. The thesis describes a novel strategy for both face tracking and facial feature detection. Face detection is important because it restricts the field of view and thus reduces the amount of computation for further face recognition or transmission, while facial feature detection is important because it enables face normalization and leads to size invariant face recognition. Resembling the way of human perceiving face and facial features, the proposed face tracker initially utilizes motion as the major cues, then human eyes are searched from the area which likely contains a human face.

The contributions of this thesis are threefold. First, we have introduced an effective system architecture including a system supervisor controlling five virtual perceptual sensor processes. Second, we have advanced a hybrid approach for face detection and facial feature extraction. An adapted Hough Transform is introduced in search of human eyes within the candidate face region. Finally, implementation issues are explored when using a low-cost Internet camera under a common Windows environment. Methods of using Microsoft Video for Windows (VFW) and real-time programming approaches are employed in this specific face tracking application.

Dedicated to my Family

for their Love and Support

Acknowledgement

I would like to acknowledge Dr. Kwan for introducing and suggesting face tracking as the research topic for my thesis, for providing me many technical papers covering the current techniques in face tracking and recognition, and for providing me with many source codes of video processing programs for this thesis work. I also benefited from his insights into all the research aspects of face tracking in this thesis through personal discussions and research seminars.

I would also like to thank Professor P.H. Alexander, Dr. B. Boufama and Dr. Indra A. Tjandra, who were my committee members, for their valuable advice and patient instruction throughout my thesis fulfillment.

Finally, I thank all my friends in ISPLab, Jie Zhang, John Qiu, Wayne Jiang, Hui Ping, Tracy Li, Halima El-Khatib for all their help, support and friendship during the study and research.

Table of Contents

List of Figures

List of Abbreviations

FOV	Field of View
API	Application Program Interface
DC	Device Context
GUI	Graphical User Interface
HT	Hough Transform

1. Introduction

1.1 Face Tracking

Machine detection and tracking of human faces from video frames is emerging as an active research area spanning several disciplines such as image processing, pattern recognition and computer vision. More and more attempts have been conducted in face localization, identification for a variety of applications including intelligent surveillance, law enforcement systems, virtual reality interfaces, etc. Most of the research has been focused on the segmentation and localization in given image(s) and the extraction of primary facial features such as eyes, eyebrows, nose and mouth.

Face detection and recognition is a difficult task mostly because of the inherent variability of the image formation process in terms of image quality and photometry, geometry, occlusion, aging, and disguise. In a video source scenario, it becomes more difficult and complicated due to constrained computing cost. Nevertheless, it does have an advantage that one can use the motion itself as a strong cue for segmenting faces of moving people.

Although over 20 years of active research on image sequence analysis is available, very little of that research has been applied to the face detection and identification problem. More and more research outcomes indicate that it is impossible to merely rely on one individual algorithm to fulfill the complicated face tracking task. A trend is to use a combination of multiple perceptual approaches

working in a complementary and supportive manner based on their mutual merits and limitations for a robust and efficient face tracking systems. The challenging and critical problem turns out to be the activation, coordination and data fusion of the candidate approaches. In real time applications like human face tracking, low level processes take majority of computing resources[6]. The efficiency of these processe's management plays an important role.

1.2Applications of Face Tracking

An automated face detection, tracking and recognition system can be utilized in many different application domains. These domains impact many aspects of human life.

In government, a face perception system can meet some of the needs in immigration border control, full-time monitoring, and airport/seaport security. It can improve criminal identification for judicial purpose and counter-terrorism techniques. This is of importance to the intelligence agencies and police departments. Defense requirements, such as military troop access control, battlefield monitoring, and military personnel authentication are applicable domains for this technique.

Detection and interpretation of a face image can also have a number of applications in a man-machine interface. One obvious use is to know whether a person is present in front of a computer screen. It is also possible to use a face recognition system as a substitute for a login password, presenting a person with his preferred workspace as soon as he appears in front of the computer system. Slightly more practical is the use of computer vision to watch the eyes and lips of a user. Eye tracking can be used to determine whether the user is looking at the computer screen

and to which part his fixation is posed. This could conceivably be used to activate the currently active window in an interface. Observing the mouth to detect lip movements can be used to detect speech acts in order to trigger a speech recognition system.

In the video-conference scenario, a passive camera with a wide angle lens provides a global view of the person seated around a conference table. An active camera provides a close-up view of whichever person is speaking. When no one is speaking, and during transitions of the close-up camera, the wide-angle camera view can be transmitted. Face detection, operating on the wide-angle images, can be used to estimate the location of conference participants around the table. When a participant speaks, the high-resolution camera can zoom onto the face of the speaker and hold the face in the center of the image. We can use an active camera to regulate zoom, pan, tilt, focus and aperture so as to keep the face of the user centered in the image at the proper size and focus, and with an appropriate light level. Keeping the face at the same place, same scale and same intensity level can dramatically reduce the information to be transmitted.

Face detection and recognition also provides a highly secure way to identify individuals because instead of verifying identity and granting access based on the possession or knowledge of cards, passwords, or tokens, verifying an identity is established using a physical characteristic. Passwords or PINs used alone are susceptible to fraud on corporate computer networks and the Internet because they can be guessed or stolen. Plastic cards, smart cards or computer token cards used alone are also not secure because they can be forged, stolen or lost, or become corrupted or unreadable. One can lose his card or forget a password, but he/she cannot as easily

lose or forget the fingers, eyes, or face. The technique of using face recognition for identification can be widely applied to ATM banking, secure communication, automated time and attendance record collection , and access control.

1.3 Thesis Objectives

The ultimate goal arises with respect to the importance of a human face detection and tracking system as a human visual perceptual substitute in a variety of applications. The thesis objective aims at the implementation of a cost effective face tracking system using a common Internet camera (Logitech Express) under Windows 98 environment. The intended tracking frame rate is 1 to 4 frames per second.

The accomplishment of this objective involves the following three major phases:

First, the strategy and architecture of the visual tracking process have to be investigated. Robustness and performance can be increased by combining multiple perceptual processes that adapt to a variety of operating conditions. As a result of using this tracking system framework with integrated visual sensors, the complexity of implementation thus turns out to be expanded by a large scale. The core component in this system will be the system supervisor controlling the perceptual processing. Its efficiency plays a critical role in the overall system performance.

In the second step, proper virtual visual sensors are selected and computationally cost effective algorithms are employed. Although numerous approaches are available in published literature, most of them are infeasible for our tracking application due to their computational complexity. In total, five visual perceptual sensors are introduced:

a motion detector, color sensor, shape analyzer, facial feature verifier and a cross correlation sensor. The system supervisor performs their activation and coordination.

The third step involves all the issues concerning the system implementation. The proposed face tracking system is to be realized in Visual C++ using Microsoft Video for Windows (VFW). The following aspects are to be handled:

- Graphical User Interface (GUI) of face tracking system with Windows
- Interfacing face tracking system with VFW to handle video frame capturing
- Computational resource and memory resource handling and their trade-off
- Display of tracking result, including intermediate information display for system demonstration and debugging.

1.4 Thesis Organization

The goal of this thesis is to advance a novel strategy and methodology for face detection and tracking from video image sequences. Related research, approaches and efforts from published literature are discussed in Chapter 2; Chapter 3 will present the system overall architecture and individual components, analysis of virtual visual sensor processes and control logic of system supervisory controller. The issues regarding the system implementation are illustrated in Chapter 4. Chapter 5 presents a detailed analysis of the system performance. Chapter 6 provides the thesis conclusions and suggestions for future research on human face tracking.

2. Related Research

2.1 Difficulties of Human Face Perception

People can easily detect and recognize familiar human faces. In general, the human recognition system utilizes a broad spectrum of stimuli obtained from many of the senses (visual, auditory, olfactory, and tactile). These stimuli are used in either an individual or collective manner for both storing and retrieval of face images. Many instances show that contextual knowledge plays an important role in face detection and hence face recognition. It is impossible (with current technology) to attempt to develop a system which will mimic all these remarkable traits of humans.

A considerable amount of published research work addresses problems and issues related to human detection and recognition of faces. Many of these studies and their findings have direct relevance to engineers interested in designing algorithms or systems for machine detection and recognition of human faces. The way infants recognize human faces also attracts interests.

There has been great interest among computer vision algorithm developers and system designers in learning how the human visual system works and in translating these mechanisms into automated systems. It is essential for a human face detection and recognition system designer to be aware that only relevant and applicable research work in psychophysics can be utilized from a practical and implementation point of view.

Automated face perception by a computer system, however, is a difficult problem since face images can vary considerably in terms of races, facial expressions, age, image quality and photometry, geometry, occlusion, and disguise.

Machine face detection and recognition is difficult also because human faces are non-rigid in shape. Face images change in a substantial scale when viewed from different direction. Even from the same direction, the human face still changes significantly with respect to face expression.

Disguises line up extra barriers for automated face detection and recognition. Glasses, beards, birthmarks and variety of hairstyle are all perplexing problems to manipulate. According to R. Chellappa in [4], different kinds of disguises must be handled differently, which requires large amount of computing resource.

2.2 Similar Face Tracking Systems

The ability to do real-time human face detection and tracking is attracting more and more research interests in the recent decade. A number of algorithms and systems of face detection and tracking have been proposed in literature.

Proposed in [9] is a system that locates and tracks the eyes of vehicle drivers to provide a driver fatigue alarm, which can potentially reduce fatigue-related accidents. This system detects driver's eyes from the first frame by using a video camera mounted on the dashboard inside a vehicle. The driver fatigue is detected by tracking eyes in subsequent frames and measuring the time when the eyes are closed.

The driver's face is first detected by finding out the vertical center line using a symmetry measuring approach, which is based on the observation that the human face

is horizontally symmetrical if upright oriented. In the second, the eye's vertical location is estimated by applying an edge detection algorithm called pixel-differentiation:

$$G(x, y) = I(x, y) - I(x-1, y) \qquad (2.1)$$

High spatial frequency areas are revealed likely to be human eyes' region. Then, the position of the iris is computed using a template composed of concentric circles.

This face tracking system used in the driver fatigue detection scenario is computationally low-cost and can be easily implemented by digital hardware. But the proposed approaches suffer from the following:

- The symmetrical measuring algorithm performs poorer when the driver's face is not vertically oriented, and exhaustedly searching each direction costs too much computational resource. This approach also often suffers from uneven lightning condition, which is very common in vehicle driving scenario.
- The proposed pixel-differentiation high spatial frequency detector can only reveal horizontal brightness variations, but high frequency components along the vertical axis have more energy then its horizontal counterparts in eye regions.

Published in [23] and sponsored by Harvard University, a hybrid face tracking system based on both sound and visual cues is realized for automated video conferencing. Initiallly, a talker is coarsely localized using a microphone array while precise localization and tracking are derived from image information.

After the coarse position of the talker is estimated from the data of the microphone array, the visual portion of the system is activated to refine the measurement and to carry out face tracking. Human bodies are first detected by applying inter-frame differencing and thresholding. A de-noising procedure is then invoked using a neighborhood-checking algorithm to remove isolated noise pixels.

An adaptive edge-tracing algorithm is used to detect the human body contour. This algorithm is based on the observation that the boundary of a human body's contour is usually smooth without sudden change. The tracking system keeps applying a relaxation algorithm until a smooth boundary is obtained which is considered to be human body contour. The neckline is determined by checking the peak concavity and convexity from the extrema of the second order derivatives of the edge contour.

Once the face has been initially detected, a tracking algorithm is activated to predict the position and velocity of the target in the subsequent frames. The video camera, which has three degrees of freedom: pan, tilt and zoom, is driven to center the detected human face in the Field of View (FOV).

Another face tracking system called CoMedi with remarkable system infrastructure is presented in [6]. CoMedi is enriched with multiple active and cooperative perceptual processes. Three visual processes (eye blink, color histogram, cross-correlation) and one audio process are use in cooperation to detect and track media-space occupants. A supervisory controller selects and controls the sequencing

of perceptual processes. As the core component, the system supervisor drives the system in a cycle composed of the following 5 phases:

- Process Selection: Based on the currently specified task and system state, the supervisor enables and disables visual processes.
- Virtual Sensors Activation: Based on the currently active set of processes, the supervisor selects proper low-level operations (or virtual sensors) to fulfil current tracking requirement.
- Visual Processes Activation: Selected visual processes are activated in sequence by the supervisor. Virtual processes are defined as a transformation from the current image to an observation vector. The observation is communicated to the supervisor and to a fusion and tracking process.
- Fusion and Tracking: Based on the results provided by the activated processes, the fusion and tracking process maintains an estimate of the center point and the size of the face using a form of recursive estimator.
- Camera Control: The current estimate of position and size of the face provides a reference signal to a servo controller for pan, tilt and zoom to maintain the media space occupant's face in the center of the FOV with desired size.

Eye blinking process, based on the difference of successive images, finds out the two small horizontally oriented regions from the thresholded difference image. The hypothesized eyes candidate regions are confirmed by checking their location with the boundaries of head movement, which is detected from frame differencing as well.

A color histogram is used to resolve face skin area from hair and ambient background. In order to divide out the luminance component, a normalized color vector is introduced and the histograms of Red and Green normalized color components are estimated for facial region extraction.

Supported by France Telecomm-CNET, CoMedi has a very efficient system architecture. But it is still suffers from computational complexity. Four Silicon Graphics Indy workstations communicating over Ethernet are used to implement the proposed face tracking system.

Published in [21], a face tracker implemented on the Datacube MaxVideo250 computer vision development platform applies a computationally intensive motion detection approach to incoming image sequences. The motion edge is obtained by applying a second order Guassian temporal function to six consecutive image frames. The motion of an edge in the image sequence then produces a temporal zero-crossing in $S(x,y,t)$ at the location of the edge in the middle frame. This approach is robust to global illumination changes and even changes in the intensity level of static objects. But it suffers from the requirement of extensive amount of memory (at least 6 frames' buffer for 'history' images) and computational cost.

A perceptual neural network trained with 1000 face images is introduced to segment the face from the motion region.

Although successful, this system is computational costly due to both its motion detection approach and its neural network perceptron. Only a 5-Hz tracking rate was achieved by using sophisticated hardware platform.

Presented in [11], a face tracker with three channels was implemented by Lucent Technology. The first of the system's three channels does a shape analysis on gray-level images to determine the location of individual facial features as well as the outlines of heads. In the second channel, the color space is analyzed with a clustering algorithm to find out areas of skin colors. The color space is first calibrated, using the results from the other channels. In the third channel, motion information is extracted from frame differences. Head outlines are determined by analyzing the shapes of areas with large motion vectors. All three channels produce lists of shapes, each marking an area of the image where a facial feature or a part of the outline of a head may be present. Combinations of such shapes are evaluated with n-gram searches to produce a list of likely head positions and the locations of facial features.

A human face tracking system presented in [2] is implemented by Microcomputer Research Lab of Intel Corporation. Intended to form part of a perceptual user interface, the algorithm used in this face tracking system is based on a robust non-parametric technique for climbing density gradients to find peak of probability distributions, which is called the mean shift algorithm. To find the mode of a color distribution within a video scene a modified mean shift algorithm is employed to deal with dynamically changing color probability distributions derived from video frame sequences. The modified algorithm is called the Continuously Adaptive Mean Shift (CAMSHIFT) algorithm. Tolerance to noise, distractors and performance is also studied. CAMSHIFT is being used as a face tracker to control games and 3D graphics.

In [5], a gaze tracking system called CapRe is implemented at the National Scientific Research Center of France to locate and track the human face and its components in order to determine the gaze of the user, who is facing a workstation. In this system, the face is first detected using an inter-frame differencing approach; the eyes and nose are then identified by searching dark zones and analyzing their constellation, finally gaze location is achieved by evaluating the orientation of eyes and pupils.

An active-camera real-time system called LAFTER is implemented by MIT for tracking, shape description, and classification of the human face and mouth using an SGI Indy computer [17]. The system is based on 2-D blob features, which are spatially-compacted clusters of pixels that are similar in terms of low-level image properties. Patterns of behavior such as facial expressions and head movements are classified in real-time using Hidden Markov Model methods. LAFTER is capable to manipulate accurate, real-time classification of the user's mouth shape without constraining head position, which makes possible real-time speech reading and expression recognition.

A critical survey of existing literature on human and machine face detection and recognition is provided in [4]. Presented in this paper are different applications of face detection and recognition in commercial and law enforcement sector. Detailed overview of more than 20 years of research conducted in the engineering community is also provided. Techniques for segmentation and localization of the face, feature

extraction are reviewed. Global transform and feature based methods using statistical, structural and neural classifiers are also summarized.

There is a trend to use integrated multiple complementary approaches to fulfil face tracking. But harmonic coordination of all the approaches and effective data fusion is the key problem in the face tracking and recognition community. More perceptual approaches can result in a more robust and reliable tracking system, but more sophisticated mechanism is needed to handle the extended complexity.

Another problem is to reduce the computational complexity and hence the cost of the entire system. Previous systems make use of either sophisticated hardware or multiple computers interconnected by LANs

3. Face Tracking System Inside-View

This chapter contains the detailed description of the face tracking system structure and the rationale of each individual system component.

3.1 Overall System Scheme

The overall face tracking system can be depicted by figure 3.1, which contains 4 levels of functional blocks including a video camera, device driver for the camera, Microsoft Video for Windows (VFW) and face tracking software. Located at the bottom level is a common Internet video camera of QuickCam(C) Express produced by LogiTech Company. The camera technical information is itemized in the following table.

Interface	USB(Universal Serial Bus)
Image	sensor CMOS
Type	Lens Manual focus
Effective pixels	Up to 352 x 288
FOV Angle	43°
Focus Range	15 cm (6 inches) to infinity
Number of colors	Up to 24 bit
Video Max frame rate	Up to 30 fps
Video resolution	Up to 352 x 288 pixels
Sensor size	1/5" diagonal
Lens Aperture	F/2.0
Distortion	Less than 10% at corners
Exposure control	Automatic

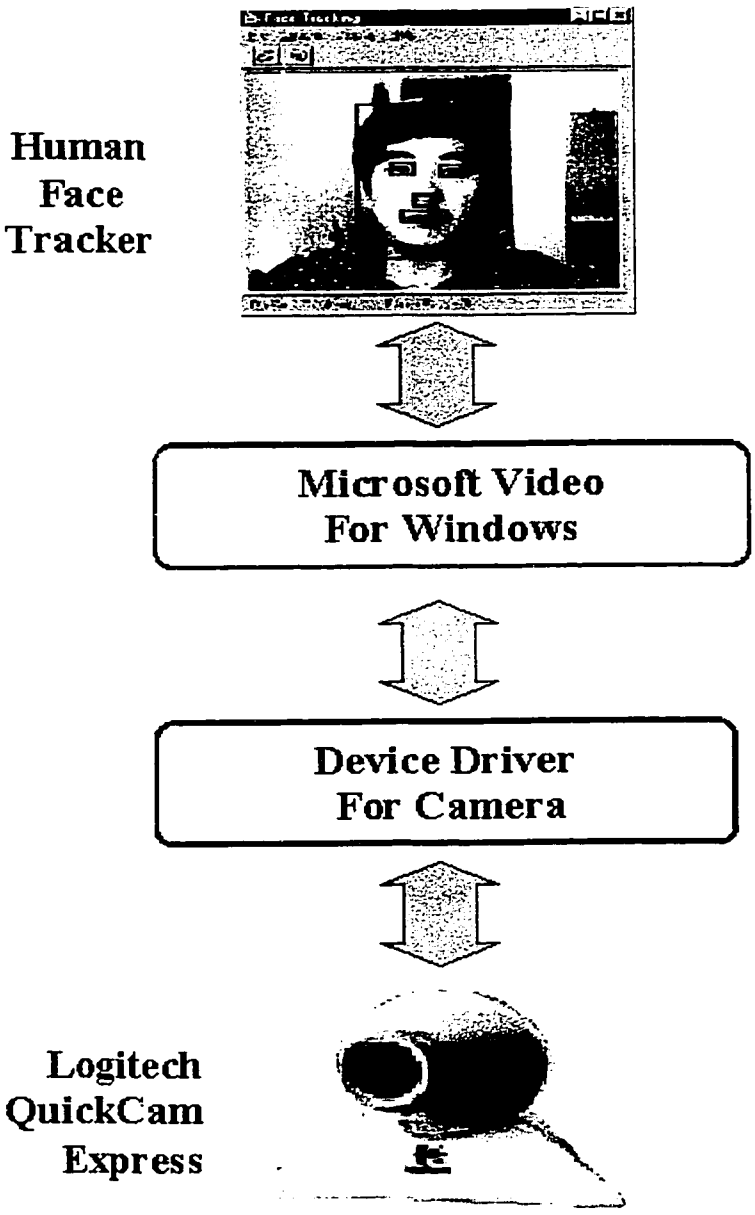


Figure 3-1 Tracking System Hierarchy

The camera device driver, which forms the second level of the system from the bottom level, provides an interface between Windows and the camera itself. Like any other peripheral equipment operating under Microsoft Windows, the camera manipulation can be programmed in a hardware-independent way. The major advantage is that the camera can be upgraded without modifying the face tracking application software. The next level of the face tracking system is the Microsoft Video for Windows (VFW), which provides functions that enable an application to process video data including the driver connection, video capture and display. The face tracking application software controls the camera and captures video frames by calling VFW functions or macros. Resembling a human with five perceptual systems controlled by the human brain, the face tracking system is developed with five virtual perceptual sensors managed by a system supervisory controller. Each virtual perceptual sensor is a process selected and activated by the system supervisor according to current system tracking demand. Not only does the face tracker topological architecture resemble the human perceptual system, but also the manner in which it perceives a human face in the FOV. This can be explained from the following two facts:

<1> The face region is hypothesized only when motion is present, which bears a resemblance to the fact that a human uses motion information to recognize another human.

<2> In the face candidate area, eyes are initially verified, which is similar to eye contact in human social activities.

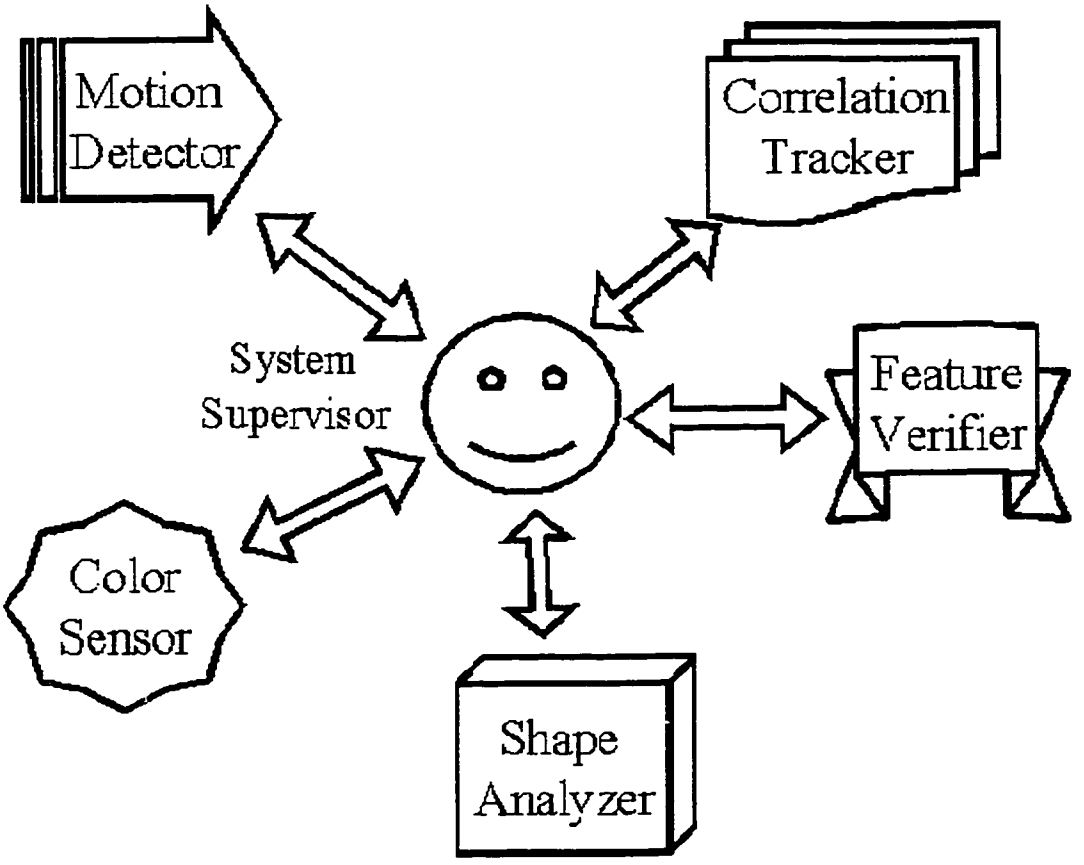


Figure 3-2 Visual Perceptual Sensors and System Super

3.2 Virtual Visual Sensor Processes

There are five virtual visual perceptual sensor processes: motion detector, color sensor, shape analyzer, feature verifier and cross correlation tracker. Robust tracking can be obtained by driving several cooperative and complementary processes, which are managed by the system supervisory controller. This section

describes each individual perceptual sensor process including their rationale, pros and cons with graphical illustrations.

Motion Detector

Human and animal visions are much more sensitive to moving objects than still ones. Many predatory animals rely heavily on motion detection to seize preys and many non-predatory animals try to avoid being victimized by staying still and using camouflages.

The face tracking system makes use of motion information as a cue to initially detect moving objects that may likely be a human body. The motion the system detects can be categorized into two types: one is common human body movement and the other is eye blinking. The reason for using eye blinking as a particular cue is because eye blinking detection is easy and reliable.

Let $I_t(x,y)$ and $I_{t+1}(x,y)$ be two consecutive frame images at time t and t+1 respectively. The difference image between these frames is given by:

$$\Delta I(x,y) = | I_{t+1}(x,y) - I_t(x,y) | \qquad (3.1)$$

The difference image $\Delta I(x,y)$ is thresholded to obtain a binary image to label moving pixels in the FOV.

$$B(x,y) = \begin{cases} 1 & \text{if } \Delta I(x,y) > T \\ 0 & \text{otherwise} \end{cases} \qquad (3.2)$$

Where T is the threshold which is empirically selected as 26.

In order to eliminate isolated noisy pixels from the binary image B(x,y), a majority filter will be utilized as following:

$$B(x,y) = \begin{cases} 1 & \text{if majority of the neighborhood of pixel } (x,y) \text{ is 1.} \\ 0 & \text{Otherwise } B(x,y)=0. \end{cases} \quad (3.3)$$

The size of the neighborhood is 3x3.

Figure 3.3 illustrates the common head motion detection result.

A human must periodically blink to keep eyes moist. Blinking is involuntary and fast. Most people do not notice when eyes blink. However, detecting a blinking pattern in an image sequence is an easy and reliable means to detect the presence of a face. Blinking provides a space-time signal, which can be easily detected and is unique to human faces. The fact that both eyes blink together provides redundancy that permits blinking to be discriminated from other motions in the scene.

If the eyes happened to be closed in one of the two images, there will be two small roundish regions over the eyes where the difference is significant. When a pair of such roundish and nearly horizontally-oriented regions are found, human eye's presence is assumed. Further perception is to be made by the Feature Verifier to find out other facial features such as nose and mouth.

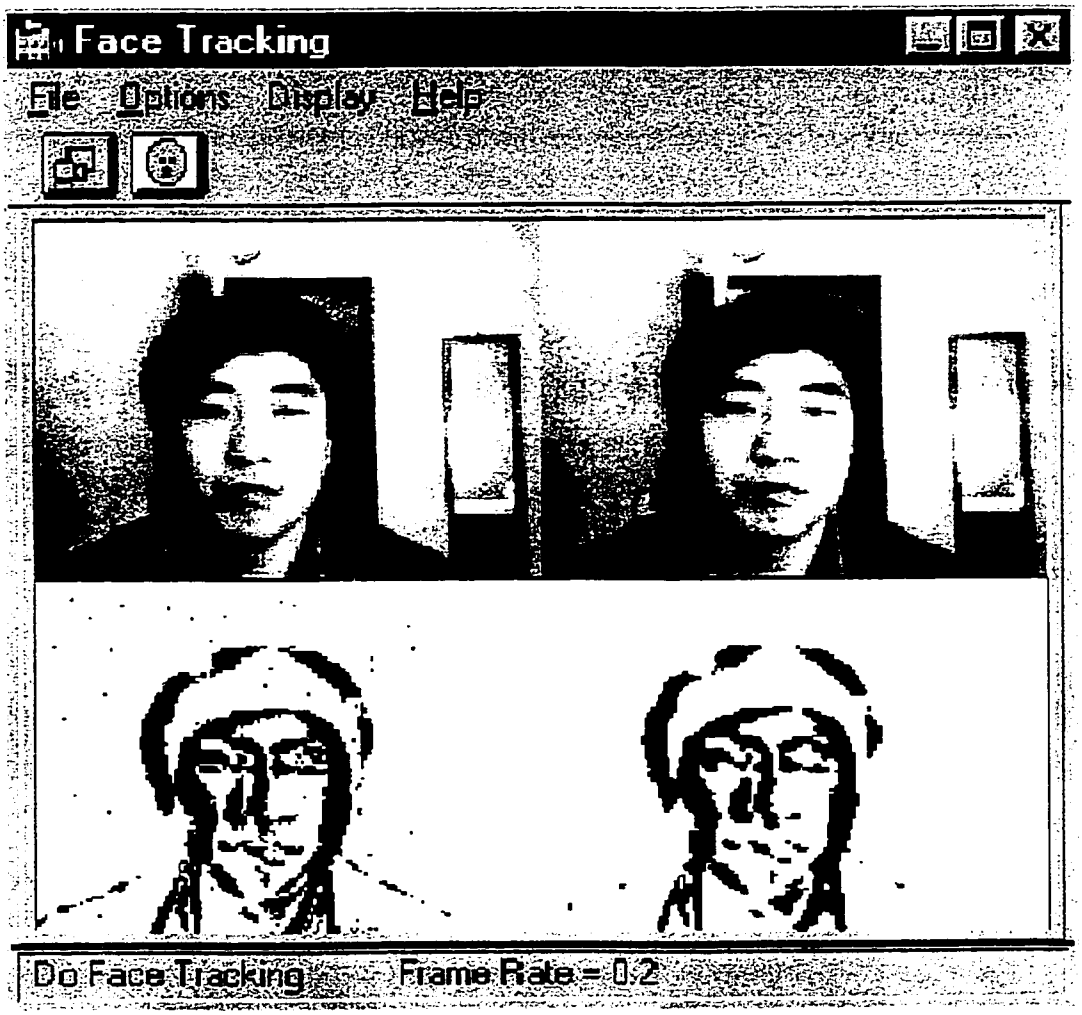


Figure 3-3 Motion Detection

Top left image ---------- First frame
Top right image -------- Second frame
Bottom left Image ----- Thresholded difference image
Bottom right image --- Majority filtered image.

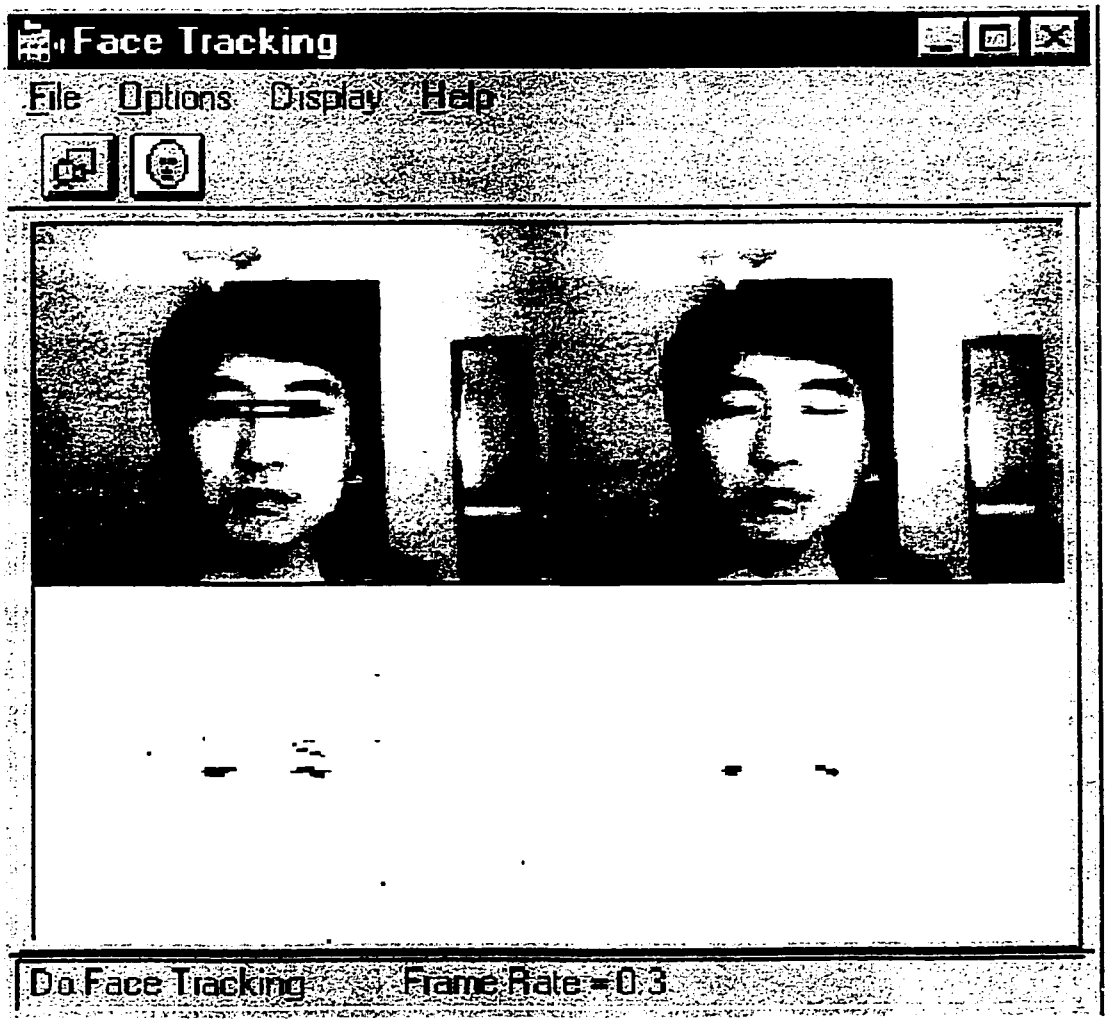


Figure 3-4 Illustration of Eye Blinking Detection

Top left image ---------- First frame with eye region mark
Top right image -------- Second frame
Bottom left Image ----- Thresholded difference image
Bottom right image --- Majority filtered image.

Although eye blinking detection is easy and reliable, it needs a sufficient size of human face to obtain a reliable result. Also eye blinking can not be easily isolated from other motions, otherwise the detection procedure will be too time-consuming.

Due to limited computational resource in the current developing environment, a high frame rate as high as the TV standard (30 Frames/s in NTSC) can not be reached. The expected frame rate is approximated at 1~4 frames/second. Because of the short period of time of eye blinking and low temporal sampling rate, the possibility to detect eye blinking is not expected to be high enough. All in all, common human movement detection is the prevailing motion technique instead of eye blinking.

Color Sensor

Color information has long been used in the face detection and recognition field. Our face tracking system introduces a color sensor as a demonstration approach but it is not used during real time face tracking mode due to its computational cost and chromatic aberration from the low quality camera.

The HSV (hue, saturation and value) color space is utilized as state in [21]. It was developed to be more "intuitive" in manipulating color and was designed to approximate the way humans perceive and interpret colors.

Figure 3. 5 illustrate the single hexcone HSV color model. The top of the hexcone corresponds to V=1, or the maximum intensity colors. The point at the base of the hexcone is black and here V=0. Complementary colors are 180° opposite one

another as measure by H, the angle around the vertical axis (V) with red at 0°. The value of S is a ratio, ranging from 0 on the center line vertical axis (V) to 1 on the sides of the hexcone.

Conversion from the RGB to HSV color space can be done as following:

Setup equations (RGB range of 0 to 1):

$M = \max(R, G, B)$

$m = \min(R, G, B)$

$r = (M - R) / (M - m)$

$g = (M - G) / (M - m)$

$b = (M - B) / (M - m)$

Value calculation:

$V = \max(R, G, B)$

Saturation calculation (saturtion range of 0 ~ 1):

If $M = 0$ then $S = 0$ and $H = 180°$

If $M \neq 0$ then $S = (M - m) / M$

Hue calculation (Hue range of 0° ~ 360°) :

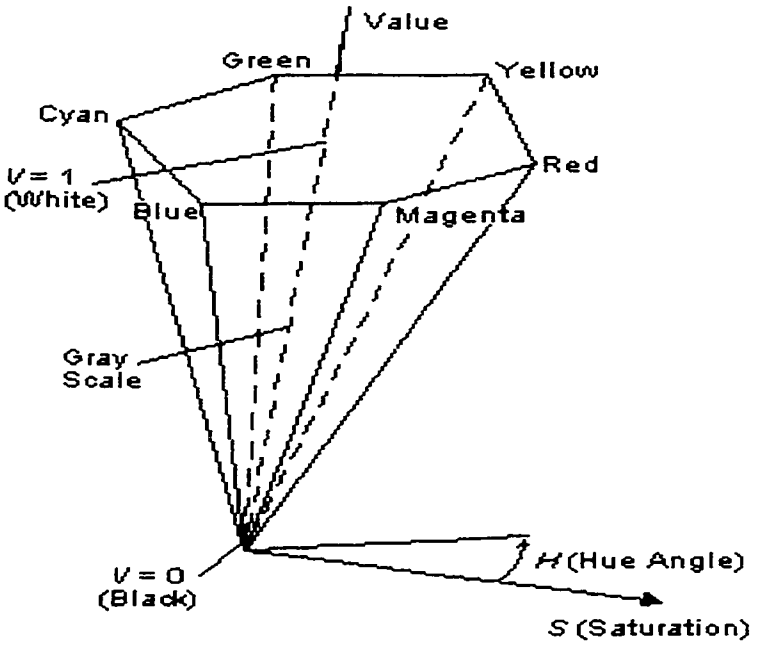
If $R = M$ then $H = 60° \bullet (b - g)$

If $G = M$ then $H = 60° \bullet (2 + r - b)$

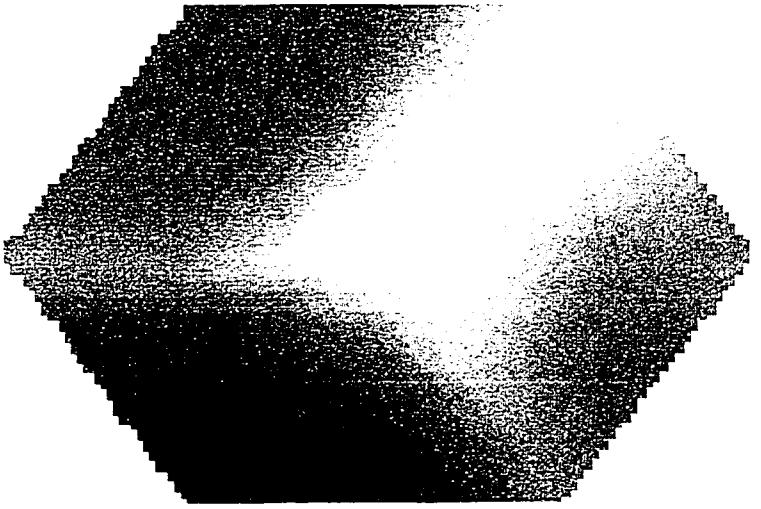
If $B = M$ then $H = 60° \bullet (4 + g - r)$

If $H > 360°$ then $H = H - 360°$

If $H < 0°$ then $H = H + 360°$



HSV Color Space



HS Slice when V=1

Figure 3-5 HSV Color Space

This approach was implemented and the face color extraction result can be illustrated as figure 3.6. From the bottom right image of with a human face, all the red tinted area indicates the extracted facial color region obtained by applying criteria as follows:

$$0.23 < S < 0.68$$
$$0^\circ < H < 50^\circ$$

The result is obviously not satisfactory. The reason can be concluded by the next two facts:

The first is that the low quality Internet video camera used by this face tracker has chromatic aberration. This can be noticed because part of the white wall turns reddish and some of the brightest pixels below the eyes turn ibluish. But the facial color sector region in HS plane does not contain any colors with highest blue component. Actually in that sector region the red component always has highest brightness.

The second is that, in our face tracking system, the color information can not play a role as important as stated in published literature in face detection and recognition community. We can undoubtedly assume that the human has utmost capability of face detection. As in Figure 3.7, When people perceive human faces, they never fail to detect a human face in a black/white image while succeed in the same image with color.

Figure 3-6 Face Region Extraction using Color Information

Top left image ---------- Original Color Image
Top right image -------- Tinted Hue Image
Bottom left Image ----- Saturation image
Bottom right image --- Face Region extraction.

Figure 3-7 Face Detection and Color Information

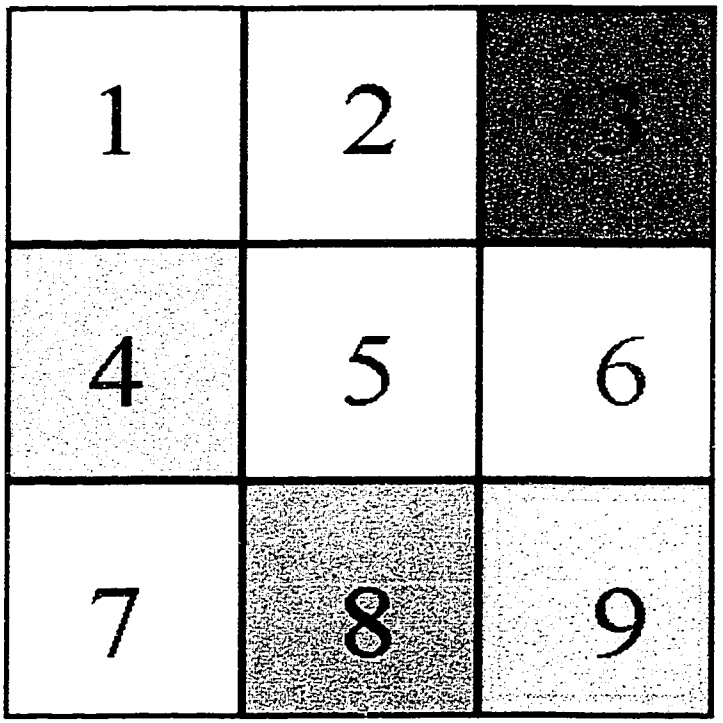


Figure 3-8 Colors from Face and Non-Face Region

Figure 3.8 shows 9 colors picked from the color image in figure 3.7. Three of them are selected from the face region while the rest from non-face regions. Since the 9 colors are confined in the uniform cubic, it is almost impossible for a human to tell face colors from non-face colors without the support of structural information.

As a matter of fact, color 1,6 and 8 in figure 3.8 are chosen from the facial region.

Shape Analyzer

A shape analyzer is employed to extract potential face regions from the binary frame image created by the motion detector. Further verifications will be performed by the Feature Verifier by means of extracting and identifying facial features. The shape analyzer applies a collection of rules to find out the regions likely to be face region. The rules also resemble manner of human to identify human faces.

The applied rules can be explained as following:

- Small and standalone regions are eliminated to avoid noisy conditions.
- Human face usually is located at relative higher part of the moving section of the image, since upright and frontal or near frontal faces are assumed.
- Shoulder like regions are absorbed by nearby higher region.
- Region near the verge of the overall image should be ignored to discard incomplete face region.

- Since eye-blink detection has higher priority, shape analyzer is skipped once eye-blink is identified. In this case, the face candidate region is generated based on the eye locations and the distance between the two eyes.

Figure 3.9 illustrates a typical video frame produced by the shape analyzer.

The tracking results prove that the shape analyzer accuracy is not very critical, since it just finds out a coarse location of potential region to avoid a time-consuming and exhaustive search performed by the feature verifier.

Shape analyzer task would not be scheduled to execute after a face is verified, since face size and location information in the previous frame can be used to determine the face region in the subsequent frame.

Figure 3-9 Shape Analyzer

Top-left Image: Binary image generated by motion detection

Bottom-left image: Silhouette image in terms of highest pixel in each column

Top-right image: Initially extracted face candidate regions.

Bottom-right image: image after shoulder region is aborbed

Facial Feature Verifier

The idea behind the Feature Verifier is to confirm the candidate region to be a face region or not by finding out human facial features such as eyes, nose and mouth. Of all the visual facial features, eyes are first to be identified, which resembles the way the human perceive human faces.

According [4], eye detection and identification are critical in human social activities. Eye contact helps to regulate the flow of communication. It also signals interest in others and increases the speaker's credibility. Eye's movement also conveys hidden and implicit meanings during interpersonal communication. A person's eyes in a static image or a video are blacked to prevent from being recognized.

In our face tracking system, human eyes are detected from the observation that the eyes' regions have relatively lower brightness than their surrounding facial skin. A computationally cost effective Hough Transform (HT) approach is introduced to detect 2 nearly vertically oriented low brightness areas within the candidate facial region.

The HT is a technique that can be used to isolate features of a particular shape within an image. The following section will illustrate how HT works to detect a straight line in an image plane.

An Example of Hough Transform Application

Consider a single isolated edge point --there could be an infinite number of lines that could pass through this point.

Each of these lines can be characterized as the solution to some particular equation. The simplest form in which to express a line is the *slope-intercept* form:

$$y = m\,x + b \qquad (3.4)$$

where m is the slope of the line and b is the y-intercept (the y value of the line when it crosses the y axis). Any line in the (x,y) plane can be characterized by these two parameters m and b.

We can characterize each of the possible lines that pass through point (x,y) as having coordinates (m,b) in some slope-intercept space. In fact, for all the lines that pass through a given point, there is a unique value of b for m:

$$b = y - m\,x \qquad (3.5)$$

The set of values (m,b) corresponding to the lines passing through point (x,y) form a line in (m,b) space. Every point (x,y) in image space corresponds to a line in parameter space (m,b) and each point in space (m,b) corresponds to a line in image space (x,y) .

The Hough transform works by letting each feature point (x,y) vote in space (m,b) for each possible line passing through it. These votes are totalled in an accumulator.

Suppose that a particular (m,b) has one vote--this means that there is a feature point through which this line passes. What if it has two votes? That would mean that two feature points lie on that line. If a position (m,b) in the accumulator has n votes, this means that n feature points lie on that line.

Generally speaking, the algorithm for the HT takes the following steps::

1. Find all the desired feature points in the image

2. For each feature points:

3. For each possibility i in the accumulator that passes through the feature point

4. Increment that position in the accumulator

5. Next possibility

6. Next feature point

7. Find the local maximum in the accumulator

8. Map each maxima in the accumulator back into image space

The slope-intercept form of a line has a problem with vertical lines: both m and b are undeterminable.

An alternative way of expressing a line is in (ρ, θ) form:

$$x \cos \theta + y \sin \theta = \rho$$

Instead of making lines in the accumulator, each feature point votes for a sinusoid of points in the accumulator. Where these sinusoids cross, there are higher accumulator values. Finding maxima in the accumulator still equates to finding the lines. The advantage of using this form is finite range for both θ and ρ. θ ranges from $-\pi$ to π, and ρ ranges in dimension of the image..

We can extend the Hough transform to find shapes of arbitrary complexity, as long as we can describe the shape with some fixed number of parameters. The number of parameters required to describe the shape determines the dimensionality of the accumulator.

An Adapted HT Approach for Eye Detection

In order to verify the existence of human faces in the candidate area, the face tracker locates human eyes based on the observance that eyes' regions have relatively lower brightness than their surroundings. A computationally cost-effective Hough transform approach is introduced to detect 2 nearly vertically oriented low brightness area.

Consider a one-dimensional function $f(x)$ with a local concave region as shown in figure 3.10. We define:

$$g(x') = min[f(x'-w), f(x'+w)] - f(x') \qquad (3.6)$$

The measurement of point x' to be a local concave can be defined as:

$$L(x') = \begin{cases} g(x') & if\ g(x') > 0 \\ 0 & Otherwise \end{cases} \qquad (3.7)$$

The measurement function for local minima search has following significances:

- Simplicity of computation, readily for hardware implementation
- Sensitivity to size of the local concave. If the size of the sunken segment is wider than 2w, it will be ruled out since min[f(x'-w), f(x'+w)] is a small value and hence the function of g(x'). Major jumps in function f(x') are also ruled out since they can as considered as a local concave with infinite width.

- Sensitivity to polarity. Local convex are to be ruled out.

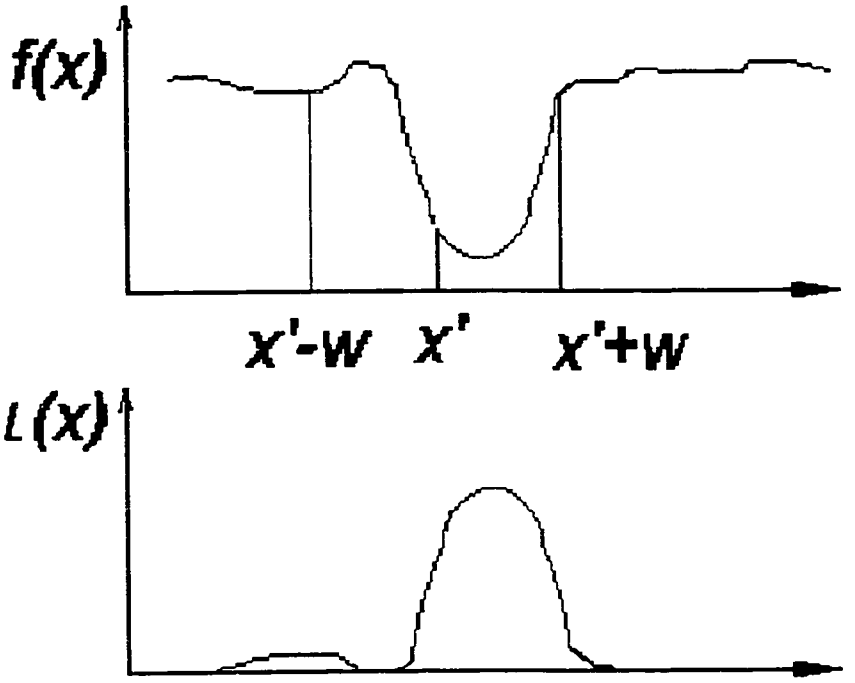


Figure 3-10 One-dimensional Function with Local Concave

Local low brightness measurement $L(y)$ along vertical direction can be estimated similarly, the only difference is shorter width due to the eyes' horizontal orientation.

Two dimensional low brightness measurements can be calculated as following:

$$L(x, y) = min[L(x), L(y)] \qquad (3.8)$$

The effectiveness this local low brightness detector can be illustrated by figure 3.8. The right image shows the extracted feature pixels from the left-hand side original image. The low brightness region such as hair is not extracted since it fails

to meet the size requirement. Only regions of eyes, nose and mouth stand out from the original image after applying the operator.

The local low brightness detector fulfills the first step of the overall Hough Transform detection for eye identification.

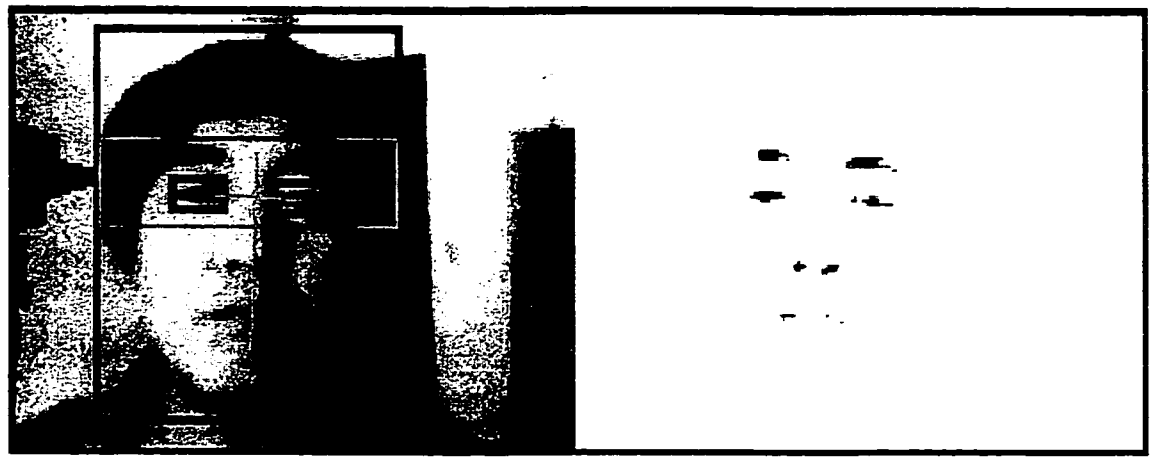


Figure 3-11 Local Low Brightness Detector

The local low brightness detector fulfills the first step of the overall Hough Transform eye detection.

The second step of the HT approach, which is feature point accumulation, is conducted by horizontally projecting the low brightness function. The projected function after rectangular filtering is illustrated by the red curve. The purpose of the rectangular filter is to merge the eye and eyebrow in this vertical projection. It is impossible to distinguish eye' location from the eyebrow's when the face tilts even in a normal degree. This problem has to be handled later after two eye's regions are separated.

The third step of the HT eye detection is very straightforward. The maximum in the projection curve is localized to determine the vertical location of the centroid of two eyes and eyebrows. The horizontal location of the centroid is calculated within the green slice shown Figure 3.8. The slice is horizontally delimited by the face candidate window (the blue window in the left-hand image), and vertically centered at the maximum position of the red projection point found out previously. The height of the green eye candidate region is proportional to the width of the blue face candidate region.

The refined individual eye position is estimated by separating the eyes and eyebrows on single eye candidate basis instead of splitting them within the whole green eye region in Figure 3.8

Face tilting is determined by the slope of the line connecting the two eyes.

A similar approach is employed to locate the nose and mouth with slight differences based on the nature of these facial features. The differences are described as following:

<1>The eyes are located within the whole face candidate region (the big blue window in Figure 3.8), while the nose and mouth are located along the perpendicular-bisector line of the straight-line connecting the detected two eye. By finding the location of the nose and mouth based on the constellation constraint of human facial features, the further search region is much smaller than the region for searching for eyes.

<2>Instead of using local low brightness region approach as the first step of HT detector, the searching for nose and mouth employs a vertical Sobel operator as a feature point detector. This is because nostrils and mouth are horizontally oriented in near-frontal faces, and also because the mouth size changes a lot when people speak.

Figure 3.12 shows a screen dump of the feature verifier of the face tracker. The bottom-right image is the original image; the top-right image illustrates the local low brightness detection algorithm. The big blue border window in the top-left image is the face candidate region obtained by the shape analyzer; the red border curve is the filtered accumulation curve; the two small blue windows show the detected two eyes. The bottom-right image shows the image after applying the Sobel operator; the cyan curve illustrates the projection of nose and mouth pixels.

Figure 3-12 Feature Verifier

Cross-Correlation Tracker

Correlation analysis has long been an effective technique to estimate the similarity between an image and a template. The definition of cross-correlation is:

$$c(u,v) = \Sigma_{x,y} f(x, y) \bullet t(x - u, y - v) \qquad (3.9)$$

It is impossible to implement (3.9) in real time without any hardware support. An alternative and less complex approach is used in this face tracking system:

$$c(u,v) = \Sigma_{x,y} | f(x, y) - t(x - u, y - v) | \qquad (3.10)$$

The template is extracted from any frame of the image sequences where a face is found and verified, which means a cross-correlation process can be started. In the subsequent images, the reference template is compared to each image neighborhood within search region.

The execution of (3.10) is 1.2 second at following image and computer configuration:

- Image size 160*120
- Template size 42*50 (Variable with face size)
- Computational resolution: every other pixel and every other line (¼ of computation of using (3.10) directly)
- Computer clock frequency 350MHz under Windows 98

The Tracking result is shown in figure 3.10, which is obtained from screen dumping of the face tracker. The top-right image is the original image; the top-left image contains a face template, which is extracted from the previous image of the frame sequence after the face is detected and verified. The size of the face is variable with respect to the face size.

The bottom-left image shows the degree of match between the original image and the template image shown in the top-left image. The darker a pixel in this image is, the better is the match at the pixel's corresponding position.

The bottom-right image, which contains a red window, indicate the position of the best match between the original image and the template.

Although the cross-correlation tracker yields a satisfactory performance, its time-consuming nature degenerates its suitability for this real time application. The 1.2-second execution time of the cross-correlation tracker alone will yield a tracking frame rate less than 0.8 frame per second.

It is implemented only for demonstration purpose.

Figure 3-13 Cross-Correlation Tracker

3.3 Intelligent System Supervisor

The entire tracking system is based on an architecture in which a supervisor activates and coordinates perceptual process. The supervisory controller selects and controls the sequencing of perceptual processes.

The control logic of the system supervisor is based on a finite state machine, in which the current state is determined by the previous state and the current system input (the active visual sensor process output in our case). The initial state of the tracking system will be the motion detection, which indicates that motion detection is the cue of face tracking based on the reality that a human usually does not remain sedentary for very long.

The system control state machine can be depicted by figure 3.14. There are five nodes in the state transition diagram. Each node represents a virtual sensor process discussed before. The current system state is the current active sensor process. Since color sensor and Cross-correlation tracker are not suitable to be used during real time tracking and only be executed in demo status, the transitions associated with them are marked with red. The blue transitions are present in both demo and real time tracking version of the tracking system.

The first node is the motion detector including eye blink detection. Motion detector is the initial active sensor process which indicates that the tracking system uses motion information as the cue for face detection.

Transitions from one sensor process to another process can be described as following:

Transition 1: When no significant motion is detected in the FOV, the system stays in motion detection state until a motion is captured.

Transition 2: If a significant motion is found and is not eye blinking, shift to shape analyzer.

Transition 3: The motion found by motion detector is not likely a person, or it is an incomplete face. This may occur when the motion is too small, too big, or close to the brink of the image.

Transition 4: This transition only happens in the demo version. After face candidate is found, pass it to color sensor for skin color checking.

Transition 5: If a face candidate region is found, apply feature verification to find out facial features.

Transition 6: Shift from color sensor state to feature verification state. Since color information can only play a subsidiary role in face detection, color detector will make no rejection of a human face. That is the reason why there is no transition from color sensor to motion detection.

Transition 7: If eye blinking is found by motion detection, a face candidate region is created and is sent to feature verifier for nose and mouth localization.

Transition 8: The face candidate region does not contain human facial features or the features do not meet the constellation constraints, reject

current session and go back to motion detection state to start a new search.

Transition 9: In demo version, once a face is confirmed by feature verifier, a face template is extracted from current location as a reference image and the system shift to cross-correlation tracking mode.

Transition 10: There is no acceptable match between the incoming image and the reference template image. This condition may be caused by face resizing, face rotating and disappearing. Motion detection is resumed for new search for human faces.

Transition 11: Matching is acceptable, stay in cross-correlation state and continue to the subsequent frame.

Transition 12: In real time tracking version, if feature verifier confirms a face, the tracking system will stay in feature verifier state. The face candidate region in the subsequent frame is to be determined by the face location in the current frame

The 12 transitions are grouped into two categories: intra-frame transitions and inter-frame transitions.

Inter-frame transitions are transition 1, 11, 12. Inter-frame transitions are performed cross two consecutive frames. A new frame capturing action has to be done to fulfil inter-frame transitions.

The rest of the transitions are all intra-frame transitions. Inter-frame transitions have to be conducted within current image frame processing session. No image capturing is needed to perform intra-frame transitions.

Similarly, states in the system transition logic are divided into tow groups: stable states and transient states. A stable state must be associated with one and only one inter-frame transition. The stable states are motion detection, feature verifier and cross-correlation tracker. Processing of an image frame must start from a stable state and terminate at a stable state.

Shape analyzer and Color sensor are transient state without a intra-frame transition connected to them. They only exist within a image frame.

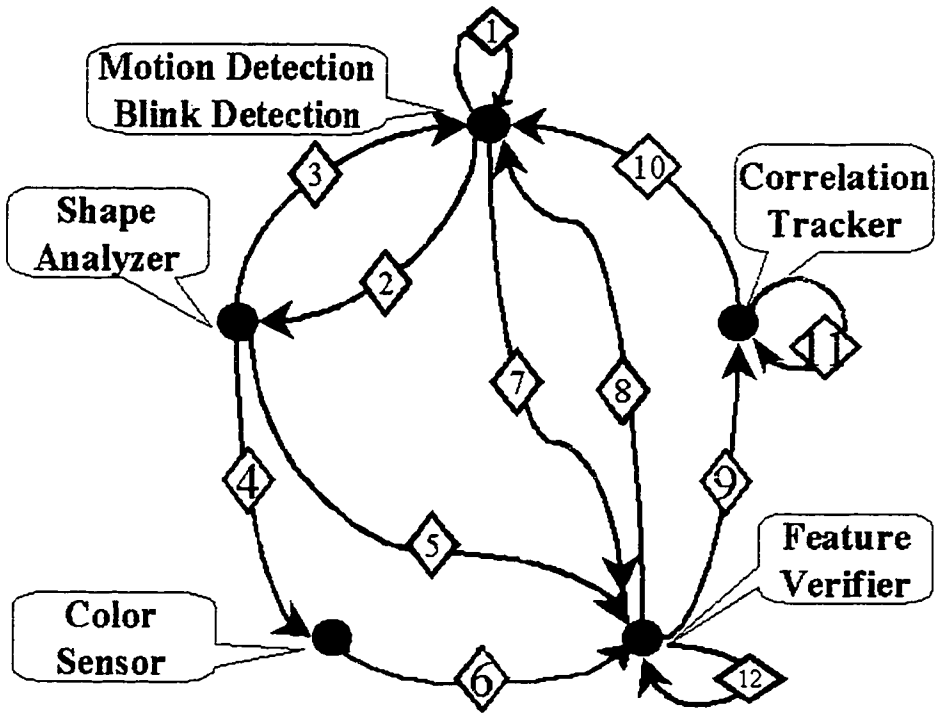


Figure 3-14 System Control Logic

4. Implement Considerations

This chapter addresses issues pertaining face tracking system implementation. The graphical interface with Windows, camera control and video capturing using Video for Windows (VFW) and real-time programming methodologies will be discussed.

4.1 Graphical User Interface with Windows

The face tracking system is a typical Windows application of Graphical User Interface (GUI). Shown in figure 4.1, the main window contains a video capture and display region, a customized menu, a toolbar of two buttons and a status bar.

Figure 4.1 Tracking System Main Window

The menu accepting user command is shown as Figure 4.2.

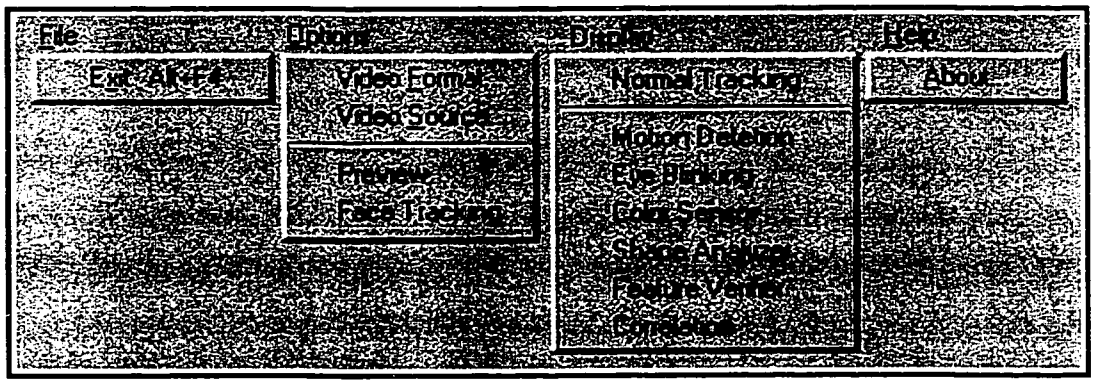


Figure 4.2
Face Tracking System Menu Command

The functionality of each menu-item can be described as following:

File:

Exit: Quit from current execution of the face tracking system

Options:

Video Format: When this command is selected, a Video format dialog box, provided by the video camera driver, is popped up to specify video stream setting. The video stream is the data format flowing from video camera to Windows. This dialog box allows user select image dimension, bit length and color format. This dialog box is unique for each capture driver. Some capture drivers might not support a Video Format dialog box. The specific video format dialog box for Logitech Express Camera is shown in Figure 4.3.

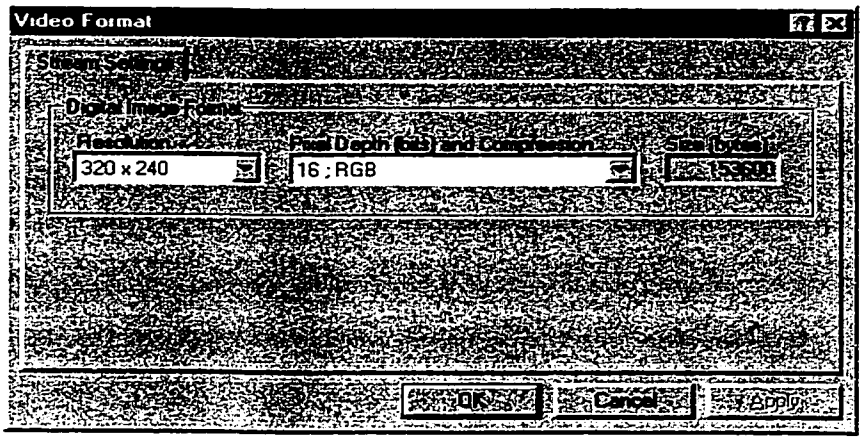


Figure 4.3
Video Format Dialog Box

Video Display: When this command is selected, a Video display dialog box, provided by the video camera driver, is popped up for the user to specify video output. This dialog box allows user to select desired illumination control from Automatic Gain Control (AGC) or manual control of the of electrical shutter exposure time and the illumination gain applied by the CCD circuits inside the camera. The image can also be flipped both horizontally and vertically. Horizontal mirror image is recommended to make the user comfortable as sitting before a glass mirror. The LogiTech Express camera also provides image luminance enhancement and color enhancement feature. One significant feature rendered by this camera is the anti flicker functionality to allow the camera be used under fluorescent light. In North America, 60 Hz should be used to coincide with the power system frequency standard.

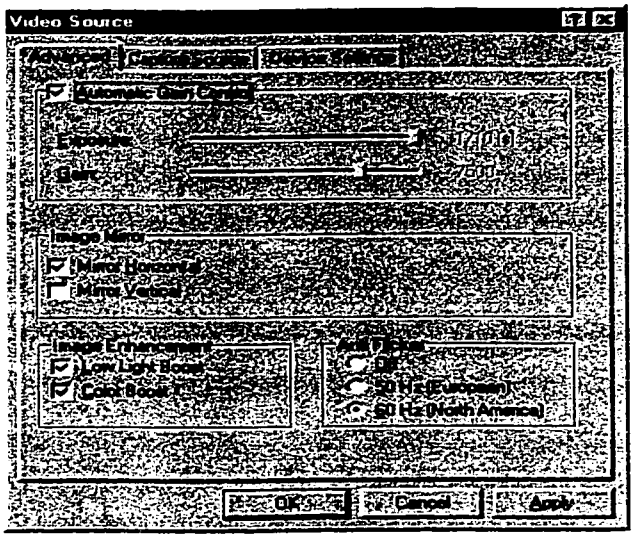


Figure 4.4 Video Display Dialog Box

Preview: When this command is selected, the face tracker will stop tracking and just continuously capture the video frame onto its main window. The 'Preview' toolbar button has the same functionality with this menu item.

Face Tracking: When this command is selected, the face tracker will activate the face tracking procedure. The tracking mode will be determined by the commands in 'Display' menu'. The 'Face Tracking' toolbar button has the same functionality with this menu item.

Display:

Normal Tracking: If this command is selected, the face tracker will be executed in the real tracking mode. The display manner is shown in figure 4.1. Any detected faces will be marked by a big red frame and facial features framed in smaller red frames.

Motion Detection: If this command is selected, the face tracker will be executed in the demo mode. The display manner is shown in figure 3.3. Motion detection information will be displayed in the image region.

Eye Blinking: If this command is selected, the face tracker will be executed in the demo mode. The display manner is shown in figure 3.4. Detected eye blinking will be displayed in a green frame.

Color Sensor: If this command is selected, the face tracker will be executed in the demo mode. The display manner is shown in figure 3.6. Color information will be displayed the image region.

Shape Analyzer: If this command is selected, the face tracker will be executed in the demo mode. The display manner is shown in figure 3.9. Shape analyzer result will be graphically displayed in the image region.

Feature Verifier: If this command is selected, the face tracker will be executed in the demo mode. The display manner is shown in figure 3.12. Hough transform feature detection result will be displayed in the image region.

Correlation: If this command is selected, the face tracker will be executed in the demo mode. The display manner is shown in figure 3.13. Template image and cross correlation image will be displayed in the image region.

Help:

About: If this command is selected, a dialog box shown in figure 4.5. The author of the software, copyright information and author's affiliation is displayed in this dialog box.

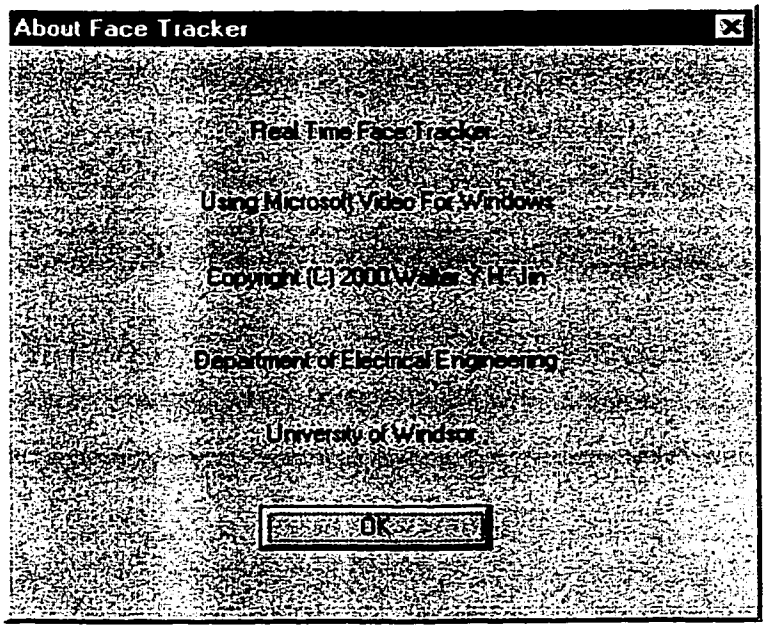


Figure 4.5 About the Face Tracker

The overall face tracking system observe the following standard steps depicted in 'C' style to fulfil its initialization:

```
int PASCAL WinMain(hInstance, HINSTANCE , LPSTR , nCmdShow)

{

    // Load a application title to display in the title bar of the main window
    LoadString(hInstance, StringID, TitleName, sizeof(TitleName));
```

```
        //Register Window class
        RegisterTrackerClasses(hInstance, hPrevInstance)

        //Create Window's object in the memory
        CreateTrackerWindows(hInstance, hPrevInstance)

        // Show the main window of the face tracker
        ShowWindow(hWndMain, TRUE) ;
        UpdateWindow(hWndMain) ;

        //Detect and Connect Camera
        DetectConnectCamera( );

        //Start Windows message loop
        while (TRUE)
        {
            TranslateMessage(msg) ;
            DispatchMessage(msg) ;
        }
}
```

The LoadString function loads a text string from the face tracker resource file, copies the string into a buffer, and appends a null-terminated character. String ID is the title string name in the resource file, while TitleName is the buffer name if the memory. The contents of this string is 'Face Tracking', which is always shown in the title bar of tracker main window.

Like any other Window application, we initially associate face tracker window procedure with a window class when registering the class. We must fill a WNDCLASS structure with information about the class. A window class is a set of

attributes that the system uses as a template to create a window. Every window is a member of a window class.

The WNDCLASS structure contains the window class attributes that are registered by the RegisterClass function. Its definition is:

```
typedef struct _WNDCLASS {
        UINT style;
        WNDPROC lpfnWndProc;
        int cbClsExtra;
        int    cbWndExtra;
        HANDLE  hInstance;
        HICON   hIcon;
        HCURSOR hCursor;
        HBRUSH  hbrBackground;
        LPCTSTR lpszMenuName;
        LPCTSTR lpszClassName;
} WNDCLASS;
```

In our face tracking system, we use the following window attribute settings:

```
TrackerRegisterClasses(HINSTANCE hPrevInst, HINSTANCE hInstance )
{
  WNDCLASS wc;
  if (! hPrevInst) {
    // If it's the first instance, register the window class
    wc.lpszClassName = TitleName ;
    wc.hInstance     = hInstance ;
    wc.lpfnWndProc   = MainWndProc ;
    wc.hCursor       = LoadCursor(NULL, IDC_ARROW) ;
    wc.hIcon         = LoadIcon(hInstance, IconName) ;
    wc.lpszMenuName  = MenuName ;
```

```
        wc.hbrBackground = GetStockObject(WHITE_BRUSH) ;
        wc.style         = CS_HREDRAW | CS_VREDRAW ;
        wc.cbClsExtra    = 0 ;
        wc.cbWndExtra    = 0 ;

        if (!RegisterClass(&wc)) {
                return(FALSE);
        }
        return(TRUE);
}
```

The above attributes specify the tracker window's icon, title name, background color, menu ID in the resource file and the style of display.

CreateTrackerWindows() function calls Window's API function CreateWindowEx ( ) to creates a pop-up, or child window with an style specified in the registered window class. CreateWindowEX( ) function sends a predefined Window's message WM_CREATE message to the window procedure.

The window created by function CreateTrackerWindows( ) is not shown in the desktop until the following functions are called

```
ShowWindow( ) ;
UpdateWindow( ) ;
```

Function ShowWindow( ) sets the specified tracker window's show state and the function UpdateWindow( ) updates the client area of the window by sending a WM_PAINT message to the window.

After the tracker window is created and displayed in the desktop, the next step is to detect and connect the video camera through VFW, which will be discussed in the subsequent section of the thesis.

As a Window application, the face tracker calls the API function TranslateMessage( ) and DispatchMessage( ) to peek messages from Windows message queue and dispatch message to the tracker window procedure.

4.2 Camera control and Video Capturing Using VFW

Microsoft Video for Windows (VFW) supports streaming video capture and single-frame capture in real-time. The face tracking system uses only a small portion of the functionality provided by VFW.

<1>The face tracker perform the following tasks with VFW:

<2>Enumerate video driver installed in a computer running Microsoft Windows

<3>Connect proper driver to LogiTech Express Camera

<4>View a live incoming video signal by using preview methods.

<5>Display video format and display dialog box provided by video driver

<6>Capture a single frame for face detection

<7>Disconnect video driver when the face tracker is commanded to close

The face tracker uses VFW function capGetDriverDescription( ) to obtain the names and versions of the installed capture drivers in a computer. Another VFW function capDriverConnect ( ) is used to connect to video driver for the Logitech QuickCam Express camera:

The following shows the example programm:

```
char    DeviceVersion[80] ;
char    DeviceAndVersion[160] ;
int uIndex, DriverCount;
for (uIndex= 0; uIndex< 10; uIndex++)
{
     if (capGetDriverDescription(uIndex,
               (LPSTR)DeviceAndVersion, sizeof(DeviceAndVersion),
               (LPSTR)DeviceVersion, sizeof(DeviceVersion)))
    {
        lstrcat (DeviceAndVersion, DeviceVersion);
        lstrcat (DeviceAndVersion, ",   ");
        DriverCount++;
        if strcmp(DeviceVersion,
                    "LogiTech USB Video Camera,  Version 1.5 .0.1596")
        {
             capDriverConnect(hWnd, uIndex)
        }
    }
}
```

The capGetDriverDescription function retrieves the version description of the capture drivers (maximum number of 10) from the list of the Windows system registry. The total number of video drivers is stored in an integer variable DriverCount. Each driver returns a version name upon the calling of this function with the right index corresponding to the driver series number in the registry.

Once a driver version of LogiTech Express camera is found, the face tracker calls another VFW function capDriverConnect ( ) to connect face application program to the camera driver. The handle of the face tracker main window is passed to the

function capDriverConnect( ) to specify that any captured frame image should be displayed to this window.

The face tracker uses VFW function capGetDriverDescription( ) to obtain the names and versions of the installed capture drivers in a computer. Another VFW function capDriverConnect ( ) is used to connect to video driver for the Logitech QuickCam Express camera:

The face tracker activates capPreview macro predefined by VFW to enable or disable preview mode.

capPreview(hWnd, TRUE);

In preview mode, video frames are transferred from the capture hardware to system memory and then displayed in the face tracker window using GDI functions. The handle of the face tracker main window is passed to the function to specify the preview window. The parameter TRUE sets the preview mode active. Before face tracking is provoked and system enters single frame capturing mode, this function should be called again with a FALSE parameter to disable the preview mode.

To activate the video format dialog box shown in figure 4.3, the face tracker uses capDlgVideoFormat ( ) macro and enables the user to select the video format. This macro needs the tracker main window handle to make the tracker window the parent window of the video format dialog box. Since the Video Format dialog box is unique for each capture driver, the dialog box shown in figure 4.3 is only valid for the specific camera of LogiTech QuickCam Express.

Similarly, to activate the video display dialog box shown in figure 4.4, the face tracker uses capDlgVideoFormat ( ) macro and enables the user adjust the video output. This dialog box is also unique for video drivers.

The following program illustrates how the face tracker uses VFW macro capGrabFrame ( ) to capture a single frame from the video driver and activate face tracking procedure.

```
CaptureAndFaceTracking( )
{
        HDC         hDC;              // Define  device context (DC)
        int         WinX, WinY;       // For the dimension of  the image
        RECT  TrackerWindpwRect;      // RECT structure for client area

        // Get client rectangle region, and its dimension
        GetWindowRect(hWnd, &TrackerWindpwRect);
        WinX = RECTWIDTH(TrackerWindpwRect);
        WinY = RECTHEIGHT(TrackerWindpwRect);

        // Disable preview mode
        vidcapSetLive(FALSE);

        // Grab a frame from video driver and display onto the tracker window
        capGrabFrame(hWnd);

        // Get device context of the tracker main window
        hDC = GetWindowDC( hWnd );

        // Start the tracker process
        DoFaceTracking( hDC, WinX, WinY);
```

```
        // Release the Device Context to Windows
        ReleaseDC (hWnd, hDC);

        // Yield Window kernel processor for 5 milliseconds
        SetTimer(hWnd, 1, 5, 0);
}
```

The Windows API function GetWindowRect is employed to retrieve the dimensions of the bounding rectangle of the face tracking window. The window dimension is represented by a Window's predefined structure RECT. Macros RECTWIDTH and RECTHEIGHT return the width and height of the window.

Macro vidcapSetLive(FALSE) is necessary to disable preview mode before the face tracker grab single frame from the driver.

Macro capGrabFrame(hWnd) captures a single frame from the video driver and display it onto the face tracker main window. The video stream transmitted from camera to the driver is activated by this macro but is hidden to the user.

The API function GetWindowDC ( ) retrieves the device context (DC) for the face tracker window. The DC handle will be passed to face tracking function to retrieve image pixel by pixel from that Device Context.

The Windows Device Context must be release to Windows because of limited DC resource.

A timer is initiated at 5 milliseconds to yield a time slot to the Windows kernel processor between two consecutive tracking frames. Without this timer, the user may feel very uncomfortable to use other applications because Windows has few time to

handle other message. Even closing a window, moving the face tracking window itself will become sluggish.

Upon the command of closing the face tracking program, a macro capDriverDisconnect (hWnd) must be executed to disconnect the driver from the face tracker. Fail to do so will stop the Windows re-activate the face tracker or any other video related application since Windows allows only one driver be active at the same time.

Real-time Programming Approaches

Since the face tracking system is a real time application, the code efficiency in term of time is critical to the overall performance. Throughout the development cycle of the system, some general rules are followed where possible.

These rules may be described as following:

Rule 1 **Try to avoid using floating point numbers as much as possible.**

Consider a multiplication operation of $x \bullet y$. It takes almost ten time longer if both operand are floating point numbers than if they are both integers. Float number calculation has to be performed by float coprocessor inside the CPU.

If a parameter has float value, we use scaled-up integer to represent it as much as possible. An example is to use an integer value to represent face tilting. A common way is using a float variable indicates the angle of face tilting in degree. We use the tangent value of the face-tilting angle, but scaled up 256 times and casted to an integer value. In this way, we got around float calculation.

Rule 2 **Try to avoid function calls as much as possible**

Calling a function in 'C' takes extra computing resource because stack pop and push of the function parameters and function returns need CPU time. The cost of not using function calls is good program architecture and program readability. But short functionality can be implemented using predefined macros. If longer function is necessary, we have to reduce the parameter list at the cost of less program safety when using more global variables.

Rule 3 **Define a union for 2-D array**

Because union in 'C' defines alias of data type, We define a union for a 2-D array and a 1-D array with the same size.

```
union TrackerUnion
{
    int image_2D[WIDTH][HEIGHT];
    int image_1D[WIDTH*HEIGHT];
} TrackerImage;
```

This is very useful to reduce computing time when clearing the 2-D image. The normal way of clearing is:

```
for ( int m = 0; m< WIDTH ; m++){
    for( int n=0; n<HEIGHT; n++){
          image_2D[m][n] = 0;
    }
}
```

A time saving way to clear the image array is to use the 1-D array:

```
for ( int m = 0; m < WIDTH * HEIGHT; m++){
    image_1D[m] = 0;
    }
}
```

The second method gets around the computing of the address of a 2-D array elements, which is an integer multiplication and integer summation for each element.

Rule 4 **Use time efficient operator**

Table 4.1 lists some time efficient statement with their less efficient statement although they yield the same results of computation

Less Efficient	Time Efficient
m + 1	m ++
2 * x	x <<2
5 * x	x<<2 + x
a * x * x +b * x + c	(a * x + b) * x + c
x = x + y	x += y

Rule 5 **Define register integer variables as array index**

Register integer variables are saved in the CPU registers instead of system memory, the CPU may access them readily. Using register integer values should be used to most frequently variables such as array index. We can not declare all the integer values as register type because of very limited number CPU registers.

Since following the rules listed above results in more programming work, they are not restricted at every line of the program. The degree of restriction varies with respect to the repeatability of code line. The following example shows the code lines with different restrict level of applying the rules:

```
// This line is lest restricted
for ( int m = 0; m< WIDTH ; m++){
    // This line is recommended
    for( int n=0; n<HEIGHT; n++){
          // This line is most restricted
    }
}
```

5. Tracking Results and Performance Analysis

This chapter of the thesis contains the discussions about the tracking system performance, including the tracking capability and timing analysis.

5.1 Tracking Capability

Figure 5.1 illustrates the face tracking system detecting a face tilted by angle 17°. When the face tilts more, the tracker starts to make false label of the left eyebrow as the left eye, which is because the let eyebrow is closer to the horizontal line pass the right eye than the left eye. This problem can be solved by investing more computing resource on larger eye vicinity to separate eyes and eyebrows at the cost of slowing down the face tracking frame rate.

Figure 5-1 Tracking Tilted Face

Figure 5-2 Tracking Small Face

Figure 5.2 shows face tracking in small face scenario. The red face frame has a width of 35 pixels out of the total image width of 160 pixels. The face tracking starts to glimmer when the face moves further away from the camera. The face tracker fails much earlier than human visions mainly because of the filters applied throughout the tracking procedure. The filters increase the robustness of the face tracking but decrease its sensitivity. The face tracker reports a presence of human face only when both eyes are found, which needs sufficient size of eyes to prevent eye pixels being filter out.

Similar situation in figure 5.3 when tracking a sided face of about 30-degree turning aside from the straight ahead position, the right eye is almost occluded and has less pixels. Also the center positions of the nose and mouth are detected with an observable missing distance.

Figure 5-3 Tracking Sided Face

5.2 Timing Analysis

In real time tracking mode, a typical tracking frequency 3.1, which means a typical face tracking takes approximately 320 millisecond. The tracking frequency varies with respect to the size of the face it is tracking and the number of active virtual sensor processes.

Figure 5.3 shows the timing analysis of a typical tracking frame. The time the face tracker spends can be split into four phases.

The first phase is the time to capture a video frame and display it onto face tracker window. Typically, it takes 110 milliseconds to capture a single frame using VFW macro capGrabFrame ( ). The internal mechanism of image capturing is unique for each individual camera and its driver. It is not controllable by user program.

In the second phase, the face tracker retrieves the captured image from the Device Context associated with the tracker window. The Windows API function GetPixel () is used to retrieve the red, green, blue (RGB) color value of the pixel at the specified coordinates. Since the execution of this API function is a slow process, to retrieve an image of 160x120 takes approximately 170 milliseconds.

The third phase is for the face tracker to detected human faces from the retrieved image. This phase normally has duration of 35 milliseconds. Since this phase takes much less time than the previous two phases, further optimization of the face tracking algorithms has little significance.

The last phase is to yield 5 milliseconds to the Windows kernel processor.

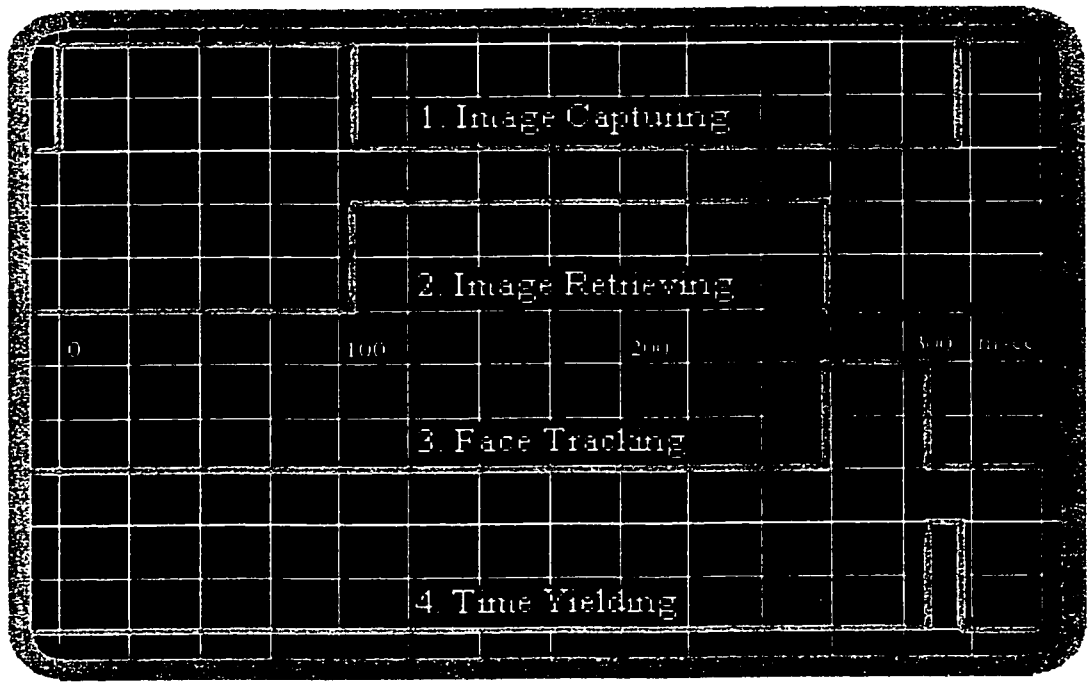


Figure 5-4 Timing Analyses

By observing the duration each phase takes, we notice that the bottleneck to speed up the face tracking is to reduce the time to retrieve an image frame from tracker window.

6. Conclusions and Future Research Work

This chapter summarizes the entire thesis and makes recommendations for future research work on real time human face tracking and detection.

6.1 Conclusions

This thesis focuses on a computationally simple and efficient, architecturally sophisticated human face tracking system implemented on an IBM 350MHz-computer running Windows 98. The system utilizes an inexpensive Internet camera as the video capturing device controlled by Microsoft Video for Windows.

The implemented human face tracking system consists of five virtual visual sensors managed by a supervisory controller, which resembles the architecture of human sensory organs controlled by the human brain. Each virtual sensor is a sensory process implemented by a 'C' software module. The system central controller organizes all the five sensor processes by following control logic described by a state machine. Resembling the manner of human visual system detecting moving, the face tracker uses motion information as the first cue. Eye blinking information is used as an auxiliary and trustworthy method to detect human face. Due to the prerequisites of sufficient size of human face and fast enough processing speed between two consecutive image frames, we do not just count on eye blink detection as the only approach to locate the face in the FOV. The shape analyzer finds out the coarse location of the moving region and excludes those regions unlikely to be a human face.

A computationally effective Hough Transform approach is originated to verify face candidate region by finding out facial features of eyes, nose and mouth.

Color information is not used in the real time tracking mode due to an insignificant role in face detection and the color aberration, but it is implemented in the demo mode.

Cross correlation is proved to be a theoretically effective approach for face tracking, but in practice, it is not suitable for this real time application without hardware support. Cross correlation tracker is also implemented in the demo mode.

The human face tracker is a real time Windows application program with a friendly Graphical User Interface. Microsoft Video for Windows is utilized for camera control and video frame capturing.

The implemented human face tracker yields satisfactory results when the frontal or near-frontal face is at least 35 pixels in width. The face tracker can tolerate a face tilting up to 17 degrees and side turning up to 30 degrees. A typical tracking frame rate of 3.1 per second is obtained using an IBM 350MHz computer.

While acknowledging the limitation imposed by its simplicity, we can still see that the face tracker can be adapted in prototyping commercialized version of products for virtual reality user interfaces, intelligent surveillance systems, law enforcement system, etc.

6.2 Suggestions for Future Research Works

The results in the thesis have proved the effectiveness of the face tracking system architecture and algorithms. Based on the implemented approaches, substantial improvement can still be achieved to build face tracking and face recognition systems with quicker, more robust and application oriented performance. The improvements can be divided into two categories.

From an academic point of view, improvements can be made by introducing more sophisticated motion detection approaches to track multiple faces. Manipulation of occlusion and distraction are highly desired in a practical vision system and should be taken into consideration in an application specific manner. The implemented system can also be improved by detecting and matching partial facial features to track side-oriented faces. Memorization of detected faces could be also a significant improvement to associate a currently tracked face with previously lost faces. Furthermore, the most important improvement is the recognition and identification of the tracked face by matching it with those stored in designated database.

From an engineering point of view, The face tracking system can be improved by using parallel approaches and methodologies such as the Intel MMX technology or a VLSI implementation. Fast image capturing is another key issue for improvement. Cross platform implementation of this system has special significance to suit broader applications.

REFERENCES

[1] O. Bernier, M. Collobert, "MULTRAK: a system for automatic multiperson localization and tracking in real-time", IEEE International Conference on Image Processing Proceedings of 1998, Part 1, p 136-140, 1998

[2] G. R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface", Microcomputer Research Lab, Santa Clara, CA, Intel Corporation, http://developer.intel.com/technology/itj/q21998/articles/art_2.htm, 2000

[3] Y.L. Cai, H.K. Kwan, "Fuzzy classification using fuzzy inference networks", IEEE Transactions on System, Man, and Cybernetics --Part B: Cybernetics, Vol. 28, No. 3, June 1998, pp. 334-347

[4] R. Chellappa, C.L. Wilson, S. Sirohey, "Human and machine recognition of faces: a survey," Proc.IEEE 83(5) pp.705-740, 1995.

[5] C. Collet, A. Finkel, G. Rachid, "CapRe: A gaze tracking system in man-machine interaction," Conference on Intelligent Engineering Systems, INES Sep 15-17 1997 Budapest, Hungary, p.p 577-581, 1997.

[6] J. Coutaz, F. Berard, J.L. Crowley, "Coordination of perceptual process for computer mediated communication," Internat. Conf. On Automation Face and Gesture Recognition, Killington, Vermont, USA, pp. 106-111, 1996.

[7] J. L. Crowley, Y. Demazeau, "Principle and Techniques for Sensor Data Fusion", signal Processing, Vol. 32. No. 1-2, pp5-27, May 1993

[8] J.L. Crowley, J.M. Bedrune, "Integration and Control of Reactive Visual Processes", 1994 European Conference on Computer Vision, p 18-25, 1994.

[9] M. Eriksson; N. P. Papanikolopoulos, "Eye-tracking for detection of driver fatigue," IEEE Conference on Intelligent Transportation Systems, ITSC Nov 9-12 1997,Boston, MA, USA pp. 314-319. 1997.

[10] V. Govindaraju, S.N. Srihari, D.B. Sher, "A computational model for face location", Proceeding of 3rd Int. Conf. on Computer Vision, pp. 718-721, 1990

[11] H.P. Graf, E. Cosatto, D. Giffpn, M. Kocheisen, E. Petajan, "Multi-modal system for locating heads and face," Internat. Conf. On Automation Face and Gesture Recognition, Killington, Vermont, USA, pp. 88-93, 1996.

[12] P.W. Hallinan, "Recognizing human eyes", SPIE Proceedings, Geometric Methods in Computer Vision, Vol. 1570, pp. 214-226, 1991

[13] J. Keith, "Video Demystified", HighText Publications Inc, USA, 1996

[14] H.K. Kwan, Y.L. Cai, B. Zhang, "Membership function learning in fuzzy classification". Int. J. Electronics, 1993, Vol. 74, No. 6, pp. 845-850.

[15] S. McKenna; S. Gong, "Tracking faces," Proceedings of the 1996 2nd International Conference on Automatic Face and Gesture Recognition, Oct 14-16, 1996 Killington, VT, USA, pp. 271-276, 1996.

[16] S. McKenna, S. Gong, "Non-intrusive person authentication for access control by visual tracking and face recognition," Proceedings of the 1st International Conference on Audio- and Video-based Biometric Person Authentication, Mar 12-14 1997, pp.177-181, 1997.

[17] N. Oliver, A. P. Pentland, "LAFTER: Lips and face real time tracker", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Proceedings of 1997, pp. 123-129, 1997

[18] J. Richard, M. Sezan, Ibrahim, E Kristine, "Robust real-time face tracking algorithm", IEEE International Conference on Image Processing Proceedings, Part 1, p 131-135, 1998

[19] S. Rougeaux, Y. Kuniyoshi, "Robust real-time tracking on an active vision head," Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Part 2 (of 3), pp. 873-879,1998.

[20] K. Sobottka, I. Pitas, "Segmentation and tracking of faces in color images," Proc. Of the Second Intl. Conf. On Auto. Face and Gesture Recognition, pp.236-241, 1996.

[21] K. Sobottka, I. Pitas, "A novel method for automatic face segmentation, facial feature extraction and tracking," Signal Processing: Image communication 12, pp. 263-281, 1998.

[22] K.K. Sung, T. Poggio, " Example-based learning for view-based human face detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, N0. 1, Jan. 1998.

[23] C. Wang, M.S. Brandstein, "A hybrid real-time face tracking system," Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP. Part 6 (of 6), May 12-15, 1998, v.6, 1998.

[24] H. Wu, Q. Chen, M. Yachida, "An application of fuzzy theory: face detection," International workshop on automatic face- and gesture recognition, Zuich, pp. 314-319, 1995.

[25] J. Yang, A. Waibel, "A real-time face tracker," Proceedings of the 1996 3rd IEEE Workshop on Applications of Computer Vision, WACV'96 Dec 2-4, 1996, Sarasota, FL, USA, pp.142-147, 1996.

[26] G. Yang, T.S. Huang, "Human face detection in a scene", Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition, pp453-458, 1993

[27] A.L. Yuille, D.S. Cohen, and P. Halliman. "Feature extraction from faces using deformable templates". International Journal of Computer Vision, 8: pp.104–109, 1992.

Vita Auctoris

Name: Yonghua Jin (Walter)

Place of Birth: Hebei, China

Education: B. Eng.
Department of Radio Engineering
ChangChun Institute of Post and Telecomm.
Jilin, China,
1979 – 1983

M. Eng
Shenyang Institute Automation
Chinese Academy of Sciences
Liaoning, China,
1985 – 1988

M. A. SC.
Electrical and Computer Engineering
University of windsor
Windsor, Ontario, Canada
1998 – 2000