COOPERATIVE WINDOWING FOR REAL-TIME VISUAL TRACKING

By

Samer Chaker Nassif, B.Sc. , M.Sc. , P. Eng.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosophy

COOPERATIVE WINDOWING FOR REAL-TIME VISUAL TRACKING

DOCTOR OF PHILOSOPHY (1997) McMASTER UNIVERSITY

(Electrical and Computer Engineering) Hamilton, Ontario

TITLE: Cooperative Windowing for Real-Time Visual Tracking.

AUTHOR: Samer Nassif, B.Sc. , M.Sc. (University of Michigan-Dearborn)

SUPERVISORS: Professors D.W. Capson and M. A. Elbestawi

NUMBER OF PAGES: xiii, 104

ABSTRACT

A new, computationally efficient windowing methodology for motion tracking is described. The proposed approach is well suited to real-time focus-of-attention applications in which regions-of-interest, or windows, are used to reduce image data rates. Applications include robot guidance, where high speed image processing is required for real-time position control in operations such as fixtureless assembly for flexible manufacturing.

A hierarchy of windowing functions which includes motion detection and target detection and tracking has been developed. This has resulted in a new algorithm for corner detection in image windows, as well as a proposal for measuring the information content of an image based on corner location accuracy. The techniques have been experimentally verified with the implementation of a vision system based on a high speed digital camera, a custom-built video interface board, and a network of digital signal processors. Dynamically positioned at video frame rates, windows within the camera field-of-view are made cooperative by exchanging information among their corresponding processors to allow real-time adaptation to visual motion. A cooperative windowing scheme using two networked target tracking windows is demonstrated. Motion tracking is based on the best-case output of the simultaneous application of a feature-based algorithm applied in the first window and a model-based algorithm running in the second. The experimental results demonstrate the advantages of motion tracking using this cooperative windows approach.

ACKNOWLEDGEMENTS

The author cannot thank God Almighty enough for the great help he has received during the period of this research work including :

-The financial support provided by McMaster University.

-The support of Dr. Capson and Dr. Elbestawi.

-The help of Dr. Vaz, who has been a source of ideas for the development of this work.

-The help of the department technician Ken Frost in the development of the hardware.

-The encouragements of friends and family to pursue this Ph.D. degree.

Finally, the love and patience shown by my parents and my wife during the good times and the bad times enabled me to complete the requirements of this degree.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

3D	-	Three dimensional
2D	-	Two dimensional
SSD	-	Sum of squared differences
rad/s	-	Radians per second
cm/s	-	Centimeters per second
Hz	-	Hertz
VME	-	Virtual memory expansion
CCD	-	Charge coupled device
FPGA	-	Field programmable gate array
DSP	-	Digital signal processor
C40	-	TMS320C40
MOPS	-	Million operations per second
CPU	-	Central processing unit
Mb/s	-	Megabytes per second
MHz	-	Megahertz
PC	-	Personal computer
FOV	-	Field-of-view
SPMF	-	Single processor, multiple function
TPDF	-	Triple processor, dedicated function

MPDF	-	Multiple processor, dedicated function
Kb	-	Kilobytes
TTL	-	Transistor transistor logic
SIMD	-	Single instruction multiple data
ATR	-	Automatic target recognition
NCC	-	Normalized cross-correlation
SSD	-	Sum of squared differences
SAD	-	Sum of absolute differences
ms	-	Millisecond
mm	-	Millimeter
GSS	-	Golden section search
TH	-	Threshold
RLT	-	Rate of loss-of-tracking
COOP	-	Cooperative windowing scheme
CD	-	Corner detector
SSE	-	Sum of squares of the errors

CHAPTER 1

INTRODUCTION

1.1 Overview

Humans can apply their sense of vision to move in 3D space, and to detect and track moving objects with great ease. However, incorporating such visual abilities in machines has proven to be a difficult task. While growing research efforts have led to major advances in this area of computer vision, more work still needs to be accomplished before machines can be given 3D motion detection and tracking abilities similar to those of humans [Huang94].

The list of applications motivating this interest in real-time 3D motion analysis has also been growing to include a wide variety of applications [Aggarwal88]. Robots with the ability to navigate freely on the factory floor and handle industrial parts at the same time, will represent a significant boost to the automation industry. The surveillance of people for reasons of security, or the detection and tracking of speeding automobiles are two examples of domestic applications. Military applications including the automatic detection and tracking of moving targets, such as tanks and warplanes, are also of great interest. The space industry is increasingly relying on visual data in the servoing of robot

arms for the placement and retrieval of satellites. These are only a few examples to indicate the diversity of applications where real-time 3D motion analysis is of critical importance.

Our application, which involves the real-time 3D motion tracking of a part being placed on a surface by a robot arm, is relevant mainly in the automation industry. However, the proposed work can also be applied in some of the above mentioned areas, i.e. domestic, space, and military.

Based on the literature survey (section 1.2.2), we have developed and implemented a hardware-based windowing system capable of the real-time acquisition and processing of various windows at frame rates (section 1.3.1). The selection of a motion tracking algorithm involved an extensive literature survey (sections 1.2.1,1.2.3) to review the different approaches currently being used. The survey shows that these approaches are mainly variations of two methods: the optical flow-based method, such as Sum-of-Squared-Differences (SSD), and the correspondence-based method, such as edge detection. To take advantage of the merits and demerits of each approach (section 1.3.2), we have developed an implementation that uses both approaches in a 'multiple cooperative windowing' scheme for a more robust tracking. A discussion of the confidence measure techniques needed to determine the tracking results is included in section 1.3.3.

1.2 Literature Survey

1.2.1 Motion Analysis Methods

The various approaches to motion analysis are generally classified in two main groups: Optical flow-based methods, and Correspondence-based methods [Vega89].

A- Optical flow-based methods

Optical flow refers to the distribution of the apparent velocities of moving brightness patterns in an image. These brightness patterns represent the moving objects in the image. The optical flow arises from the relative motion of the objects and the imaging sensor. Optical flow can provide important information about moving objects including their spatial arrangements and structural features. Discontinuities in optical flow can also be used in segmenting images into regions that correspond to different objects [Vega89].

The optical flow constraint equations can be derived using different methods [Horn81], [Schunk84]. The method proposed by [Horn81] assumes that the image brightness varies smoothly without any spatial discontinuities, which ensures that the brightness function is differentiable. [Schunk84] derives the optical flow constraint equation using a different approach, which assumes that the perceived change in image

brightness is entirely the result of translational motion, and that the image is smooth, except for a finite number of lines of step discontinuities. A description of other approaches for the estimation of optical flow and the relations between them is also given in [Nagel87].

While optical flow-based algorithms are in general computationally fast, they are usually based on some assumptions which are hard to obtain. The computation of the optical flow also requires the evaluation of partial derivatives of image brightness values. Since the evaluation of derivatives is a noise enhancing operation, it can have an adverse effect on the estimation of the optical flow. Therefore, the images have to be subject to some processing before starting the motion computation.

B- Correspondence-based methods

Correspondence is the process which identifies elements in different views as representing the same object at different times, therefore maintaining the perceptual identity of objects in motion or change [Thom79]. The correspondence problem can be tackled at the level of feature points, surfaces, or whole objects. However, establishing and maintaining such correspondences is not a trivial task. The development of robust techniques to solve the correspondence problem is still an active area of research. Current approaches include cross-correlation-based methods ([Arking78], [Tian84]), and feature-based methods ([Barnard80], [Shah84], [Sethi87]).

The cross-correlation of two images can be used to search for an object, and to determine the object's relative displacement from one image to the next. [Arking78] applies a cross-correlation-based method to measure cloud motion from satellite imagery. [Tian84] recursively computes a new estimate of the object's position based on the peak of the cross-correlation. The sharper the peak, the more reliable the motion estimation becomes.

In [Barnard80], a technique for matching features in stereo imagery based on smoothness in change of depth is proposed. The same method can be applied to match features in two monocular images, based on smoothness in spatial displacement of image features. [Shah84] discusses the use of a measure of cornerness to identify and track the motion of objects. Corners are generally used because the two velocity components at these feature points can be easily computed. A method for finding and maintaining correspondence between feature points based on a long sequence of monocular images is presented in [Sethi87]. The iterative optimization algorithms used are based on preserving the smoothness of velocity changes.

The above examples illustrate some approaches used in correspondence-based methods for the computation of motion. Although several methods and approaches have been developed to solve the correspondence problem, it is still a difficult task with reliable solutions only in the case of constrained applications. Prior image segmentation and feature labelling are usually required before a unique solution for object displacement

can be determined. In general, correspondence-based methods are more computationally expensive, but less sensitive to noise.

1.2.2 Vision System Requirements

The vision systems used in 3D motion tracking applications can be based on using one camera, multiple cameras, or even no cameras, by using different sensors such as range-finders instead.

A- Range-Finders

Little work has been done so far in the area of range-finder sensing for real-time motion tracking applications [Arch88], and [Venkat90]. The advantage of using range-finders over cameras being that depth information is immediately available. Although in general cameras provide more reliable information than range-finders at a lower power consumption.

In [Venkat90], two laser range-finders are mounted on the wrist of a robot for the tracking of a flat object in five degrees of freedom. Sensory feedback from the range-finders is used to servo the robot, so that it can maintain a trajectory similar to that of the object. The Maximum linear and angular tracking speeds are reported to be 25 cm/s and .5 rad/s respectively. However, the robot maintains the required pose at these speeds only

with a small time lag.

B- Multiple Cameras (Stereo Vision)

Stereo vision is a technique for determining the 3D description of a scene observed from several viewpoints. It is often used in the recovery of the depth and 3D motion of moving objects. While the term stereo vision immediately brings to mind the use of two cameras (binocular vision), many stereo vision systems have been developed based on three cameras (trinocular vision), or even more than three cameras.

In [Dhond89], a review of major stereo algorithms, which use binocular images, is presented. The three main stages in stereo analysis are preprocessing, establishing correspondence, and 3D depth computation. However, solving the correspondence problem is difficult, mainly because the geometric constraints of binocular stereo are not sufficient to impose a unique solution. Several heuristic constraints must be added before an adequate solution can be computed [Ayache91]. In the recent past, new techniques for achieving matching based on trinocular imaging have been proposed ([Ayache87], [Ito86], [Pietikan86], and [Yashida85]). A cost-benefit analysis of adding a third camera for stereo correspondence has been conducted by [Dhond91]. The results show that trinocular matching reduced the percentage of mismatches by more than one half when compared to binocular matching. On the other hand, the trinocular stereo analysis increased the computational cost by about one forth over the binocular analysis.

Gennery et *al.* [Gennery87] describe the research at JPL involving a space telerobot with a perception subsystem using five video cameras, which can provide the locations, orientation, and velocities of objects in the work environment. Special image processing hardware (PIFEX) is used to maintain real-time operations.

Although using stereo vision is advantageous in recovering 3D information, the computational cost is high, and special hardware may be needed to achieve real-time performance is achieved.

C- One Camera (Monocular Vision)

Monocular vision has mainly been used in visual tracking applications involving targets moving on a plane. It is well known that an image represents a 2D projection of a 3D scene at an instant of time. In order to recover the lost information, some assumptions about the world must be made, or a sequence of image frames must be analyzed [Sethi87]. In [Silven93], the results of 3D visual tracking experiments based on monocular vision are presented. The initial 3D object position is assumed to be approximately known, and the tracking error is reduced through the integration of new observations. However, model uncertainties are shown to cause failure in tracking.

A major advantage in using monocular vision over stereo vision in motion tracking applications is in the computational cost. While the recovery of 3D information using

monocular vision is generally a difficult task, application specific assumptions can be made to reduce the level of complexity [Pap91].

1.2.3 Motion Analysis Processing Requirements

Real-time 3D vision applications involve several key issues which must be considered in the design of the vision system, if an adequate performance is expected. The first issue to be considered is the motion analysis process.

In general, motion analysis can be divided into three main processes. The first is an early detection process involving low-level processing algorithms, such as image differencing, to detect motion in the early stages of image analysis [Jain81]. The second is a peripheral process in which image data is translated into symbolic data to achieve motion tracking. The final process involves high-level processing techniques in which symbolic data is manipulated to obtain recognition results. A significant number of complex operations can take place in each process, and therefore, the performance of the vision system is not expected to be satisfactory when all three processes are run sequentially on one computer.

The second important issue to be taken into consideration is that of image acquisition. The different processors of the vision system should have independent access to any area within the image frame, in order to keep the image acquisition overhead as

small as possible. Many vision systems have implemented the region-of-interest ("window") acquisition process by software ([Allen93],[Buttazzo94],[Koivo91a]), i.e. by acquiring the whole image frame, then extracting the window from it. This of course introduces unnecessary delays in the image acquisition cycle, which can only be avoided by using the proper hardware in the system, so that only the pixel data within the window is acquired. Lee [Lee92] and Lang [Lang87] use custom built cameras with on-board A/D converters and computers to acquire and process regions of interests within the image frame. The multiple window vision systems proposed by Graefe [Graefe84], and Inoue and Mizoguchi [Inoue85], include a common video bus for all of the window processors. However, the three-level hierarchy of motion analysis is not taken into consideration, which limits the performance of these systems in applications involving time-consuming algorithms. Kubota et *al.* [Kubota93] propose a multi-stage vision processor with an overall image processing unit for locating candidate regions of moving objects, a local multiprocessor system consisting of 16 modules for tracking regions of interest, and a host workstation for recognition results. For optimum performance, the communication delays between processors should be minimum, the number of processors performing any one of the motion analysis tasks should not be restricted, and the position updates should be directly accessible, so that delays associated with the use of the communication bus of the host computer are avoided.

1.2.4 Visual Servoing Applications

Vision is a useful robotic sensor fundamental to increasing the versatility and application domain of robots. Typically, visual sensing and manipulation are combined in an open-loop fashion ('look' then 'move'). More recently, machine vision has been used to provide closed-loop position control for a robot end-effector to improve its accuracy. This is referred to as visual servoing [Hill79]. Proposed applications span the manufacturing, military, and space industries. A comprehensive review of the literature in this field is given by [Corke93].

Visual servoing is the fusion of results from many elemental areas including high speed image processing, kinematics, dynamics, control theory, and real-time computing [Hager96]. Irrespective of the control approach used, the vision system is required to extract the information needed to perform the servoing task. For the purposes of our work, the high speed image processing area will be the highlight of the following literature review.

Visual servoing pre-supposes the solution to a set of potentially difficult static and dynamic vision problems. Many reported applications have assumed a simple vision problem, by painting objects, using artificial targets, or using task specific clues ([Horn87], [Castano94], [Allen93]). In less structured conditions, vision algorithms typically rely on the extraction of sharp contrast changes (edge/corner detection) to

indicate object boundaries, or on the observation that the appearance of small regions in an image sequence changes very little (SSD approach).

The implementation of an edge extraction method, using a Sun Sparc II workstation, which can localize and track up to 22 edge segments at a rate of 30 Hz is discussed in [Hager96]. However, this edge-detection scheme is susceptible to mistracking caused by background or foreground occluding edges.

[Weiss87] proposes an adaptive model-reference controller for a visual feedback system. Only simulation studies of two and three degrees of freedom systems are performed, with a highly structured environment assumed in order to keep the vision processing relatively simple.

An example of a vision system module and a feature-based trajectory generator for tracking a moving planar object is presented in [Feddema89]. The vision module uses the location of binary image features to control the position and one degree of orientation of the robot manipulator. The desired image features of the moving object must be taught to the system before tracking begins. The approximate positions of the image features are also assumed to be known initially. A steady state position error is shown to occur due to the time delay caused by the image processing unit. The vision hardware consists of a Sun 3/160 workstation, an Imaging Technologies ITEX-151 image processing hardware with a VME bus interface, and a Pulnix CCD camera.

Visual information obtained from a stationary camera is incorporated in an adaptive self-tuning controller to allow a robotic manipulator to grasp a moving object in a 2D plane [Koivo91]. Time delays due to the processing of images is reduced by selecting one out of every 8x8 pixel array, thus reducing the image size from 512x512 to 64x64. An experimental threshold value is chosen to create a binary image before processing begins and motion information is extracted. The motion of the object is assumed to be smooth, and its maximum velocity is about one third that of the robot gripper. The vision system involves a VAX 11/780 computer, a Sun workstation, and an ITEX 151 imaging system connected to the workstation through a VME bus.

Papanikolopoulos and Khosla [Pap91] address the problem of robotic visual tracking (eye-in-hand configuration) of a target that moves in 3D with translational motion. The relative motion of the target with respect to the camera is measured by using the SSD optical flow technique. The user initially selects the object features that must be tracked, and multiple 10x10 windows are then used to maintain tracking. Large tracking errors are noted when abrupt changes in trajectories occur. In addition, a larger tracking error appears in the Z direction. An extension of this work is presented in [Pap93]. The problem of visual tracking in 2D space is formulated as a combination of control and vision. The formulation is with respect to the camera and not the world frame, for better control of the camera. It is claimed that noisy measurements from the camera when combined with the control law can yield a better performance. Four 10x10 windows are placed on selected feature points of the object to track its motion. The best tracking

measurement is chosen based on the window with the best confidence measure.

The work of Wang and Wilson [Wang91] involves estimating the 3D pose of an arbitrary moving object for real-time robot tracking control. It represents an extension of the planar motion tracking control approach of [Wilson88] for estimating 3D motion parameters for 3D tracking control using Kalman filtering. The validity of this method is verified by computer simulation and real-time experiments. Because the locations and the number of object features affect the accuracy of the Kalman estimates, five non-coplanar features are used to improve the tracking performance.

A robotic system capable of intercepting and grasping a moving object based on the visual feedback from a pair of stationary cameras is presented in [Allen93]. the 3D motion parameters are computed based on the optical flow method. In their experimental results using the PUMA 560 robot and a special parallel image processing computer (PIPE), it is shown that the robot can track, intercept, and grasp a train moving in an oval path at velocities of up to 30 cm/s. The system is able to cope with sources of noise and error by applying parametrized filters that can smooth and predict the position of the moving object.

1.3 The Structure of the Thesis

In this thesis, the following important issues are addressed in order to implement

a robust real-time visual tracking system :

-The hardware.

-The motion tracking algorithm.

-The window confidence measure.

1.3.1 Hardware

Our strategy in designing a vision system with real-time performance is to minimize the delays associated with both the image acquisition and image analysis cycles. This vision system, which can implement the three processes of motion analysis in parallel, is designed based on a high speed digital camera, programmable gate array technology, and a network of digital signal processor (DSP) modules. The system is capable of acquiring and processing regions of interest ("windows") on a frame-by-frame basis. These multiple windows of varying sizes can be used as "tracking windows" for focus of attention, or as "watch windows" for peripheral vision. Unlike the vision systems proposed in [Fukui92] and [Kubota93], this system is flexible enough so that each DSP module may be assigned either one of the two window functions depending on the requirements of the application. The DSP module performing the cognitive process can also be used for dynamic servoing of a robot position controller using the DSP's external communication ports.

Multiple hierarchical windowing strategies, which implement the three processes of motion analysis in parallel, have been developed and applied, so that the real-time motion tracking capability of this system is demonstrated.

1.3.2 Motion Tracking Algorithm

The selection of an appropriate tracking method is application dependent. For example, in the case of tracking a single pattern that is approximately planar and moving at moderate speeds, the cross-correlation approach is suitable. It can accommodate some image distortions, and it can be implemented to run at frame rates for tracking small motions. Because it relies on gray value arithmetic, it is sensitive to illumination and background changes and occlusions. Thus, if a task requires the tracking of an object with occluding parts in a changing background, feature-based methods should be faster and more robust. A new computationally efficient corner detection algorithm has been developed and applied for this purpose. However, feature-based methods, which typically involve edge detection, can be susceptible to mistracking due to background or foreground occluding edges [Hager96]. Therefore, in more realistic situations, neither approach will yield a robust performance. Since the disadvantages of both methods manifest themselves in opposite scenarios, we have integrated the two approaches in two separate but networked foveal windows ('Cooperative' windows) to improve the tracking performance and achieve the desired robustness. A constant communication link between the two window processors ensures that the windows are acquired at the same instant of time,

based on the same image coordinates. The tracking results of both windows is compared and weighed using the measure of confidence of each window, before a decision is made and the window coordinates are updated.

1.3.3 Window Confidence Measure

The choice of an appropriate confidence measure for both foveal windows will have a significant effect in determining the tracking results. [Anandan87] developed a confidence measure which can recognize errors due to homogeneous areas and occlusion boundaries. The problem with this confidence measure is that it is based on the computation of second order derivatives, and therefore it is noise sensitive. [Matties89] computes the variance in the estimate of one-dimensional displacement. The computation is based on a parabolic fit to the SSD curve. In [Pap93], an extension of this technique to 2D displacement is proposed. The confidence measure statistically describes the sharpness of the minimum of the SSD curve. This technique can be computationally expensive. The applied confidence measure to implement the cooperative windowing approach is based on least-squares regression, which is not a computational burden.

1.4 Contributions of this Thesis

This work is relevant in the area of real-time motion analysis. Based on the literature survey, our work stands out with several contributions :

1- The development and implementation of the hardware-based window acquisition and processing system using off-the-shelf components.

2- The development and implementation of a hierarchical windowing method in which motion tracking is performed in real-time.

3- The development and implementation of a new corner detection method.

4- The development of a new information content measure based on the number of edge pixels in the image window.

5- The development of a cooperative windowing method in which the motion tracking results from different approaches can be combined to improve tracking.

6- The implementation of a real-time 'Cooperative Windowing' scheme, and its application to a real life experiment.

The implications of this work in the area of real-time motion analysis are significant. The real-time capability of our platform is demonstrated without the use of expensive special image processing hardware. A new corner detection approach has been proposed. Finally, the validity of integrating the results of different approaches to improve motion tracking has been established.

CHAPTER 2

A DSP-BASED WINDOWING NETWORK

2.1 Introduction

In recent years, many proposed vision systems have implemented the concept of region-of-interest (window) in visual tracking applications, to achieve real-time performance. By selective positioning of windows within the input image, the amount of pixel data to be processed can be substantially reduced. Although many systems have implemented the window acquisition process by software ([Buttazzo94], [Rizzi92], [Pap93]), this typically results in whole image frames being transferred before the image processing cycle can even begin. Hardware implementations allow for the processing to commence immediately at the completion of a window acquisition time. For example, Lang [Lang87] and Lee [Lee92] use custom built cameras with on board A/D converters and computers to acquire and process regions of interest within the image frame. The multiple window vision systems proposed by Graefe [Graefe84], Inoue and Mizogushi [Inoue85], and Kubota *et al.* [Kubota93] include a common video bus for all of the window processors. The architecture proposed in this work is based on a high speed digital camera, field programmable gate arrays (FPGA), and a network of TMS320C40 digital signal processor modules (Figure 2.1). In this system, each processor has

independent access to any area within the image frame, in order to keep the image acquisition overhead as small as possible. Multiple windows within the same image can be acquired using an efficient implementation of gate array-based custom logic that feeds only the pixel data within the designated window to a dedicated DSP. The image analysis process then begins as soon as the window data transmission is complete. The DSP can update the size and the position of the window on a frame-by-frame basis, and also communicate with other DSPs dedicated to additional windows. Windows may also overlap to any extent. The system is modular, independent of the host computer, and may be readily extended to include any number of windows.

2.2 Vision System Implementation

2.2.1 Hardware

The vision system includes a high-speed digital camera, a custom built video interface board using FPGAs for window logic operations, and a DSP-based processing board (Figure 2.1). The digital camera is a Dalsa CA-D1 camera with a CCD image sensor, an 8-bit pixel array size of 256x256, and a frame rate of up to 200 frames per second. The video interface board consists of the XILINX XC3042-125 Field Programmable Gate Arrays chips and assorted buffers. The DSP modules in the processing board are based on the Texas Instruments' TMS320C40 DSP chip which features a 275 MOPS CPU and six 20 Mb/s communication ports. The host computer

system is a 486 SX/50 MHz PC.

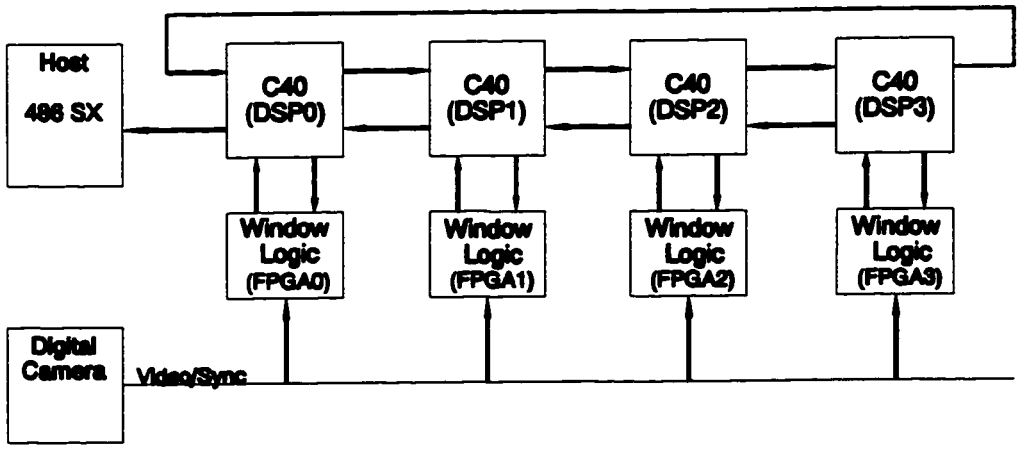


Figure 2.1. System architecture.

Our design uses two of the C40's six communication ports in the window acquisition process (Figure 2.2). One port outputs the window coordinates to the FPGA, while the other port receives the camera data through the buffers on the interface board. Two conditions must be met for a pixel to be strobed into the C40's port:

1. A valid set of coordinates (xmin,xmax,ymin,ymax) defining the window location within the camera field-of-view (FOV) must have been transmitted.
2. The pixel must be within the window.

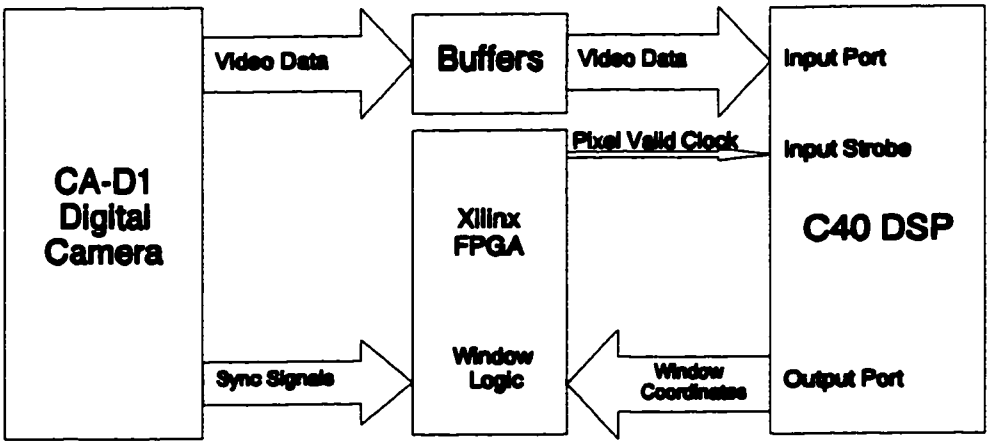


Figure 2.2. Window acquisition process.

A set of window coordinates is valid if :

$(0 \leq xmin < 255) \wedge (0 < xmax \leq 255) \wedge (xmin < xmax)$, and

$(1 \leq ymin < 256) \wedge (1 < ymax \leq 256) \wedge (ymin < ymax)$.

This difference in the ranges of the X and Y coordinates is addressed in section 2.2.2. Only one window is received for every set of window coordinates transmitted. The C40 can start processing image data as soon as window transmission is complete. A detailed

block diagram of this window acquisition process is given in Figure 2.3, and the definitions of the internal signals are included in table 2.1.

2.2.2 Window Logic Design

The window logic is implemented using the XILINX XC3042-125 programmable gate array chip, and it is divided into the following sections:

-Input : C40 Data Input and Camera Synchronization Signals

-Position Counters

-Magnitude Comparators

-Output : Strobe Generation

The C40 data input section consists of a series of four 8-bit wide D-type registers which can store one set of window coordinates. The position counters use the camera synchronization signals to keep track of the X and Y position of the current pixel relative to the image frame. The magnitude comparators determine if the current pixel is within the window that is being sampled. If it is, then a strobe signal is generated and the pixel is transmitted to the C40. A detailed description of this logic is shown in figures A1.1 and A1.2 (appendix A), and the definitions of all the internal signals in these figures are given in tables A1.1 and A1.2.

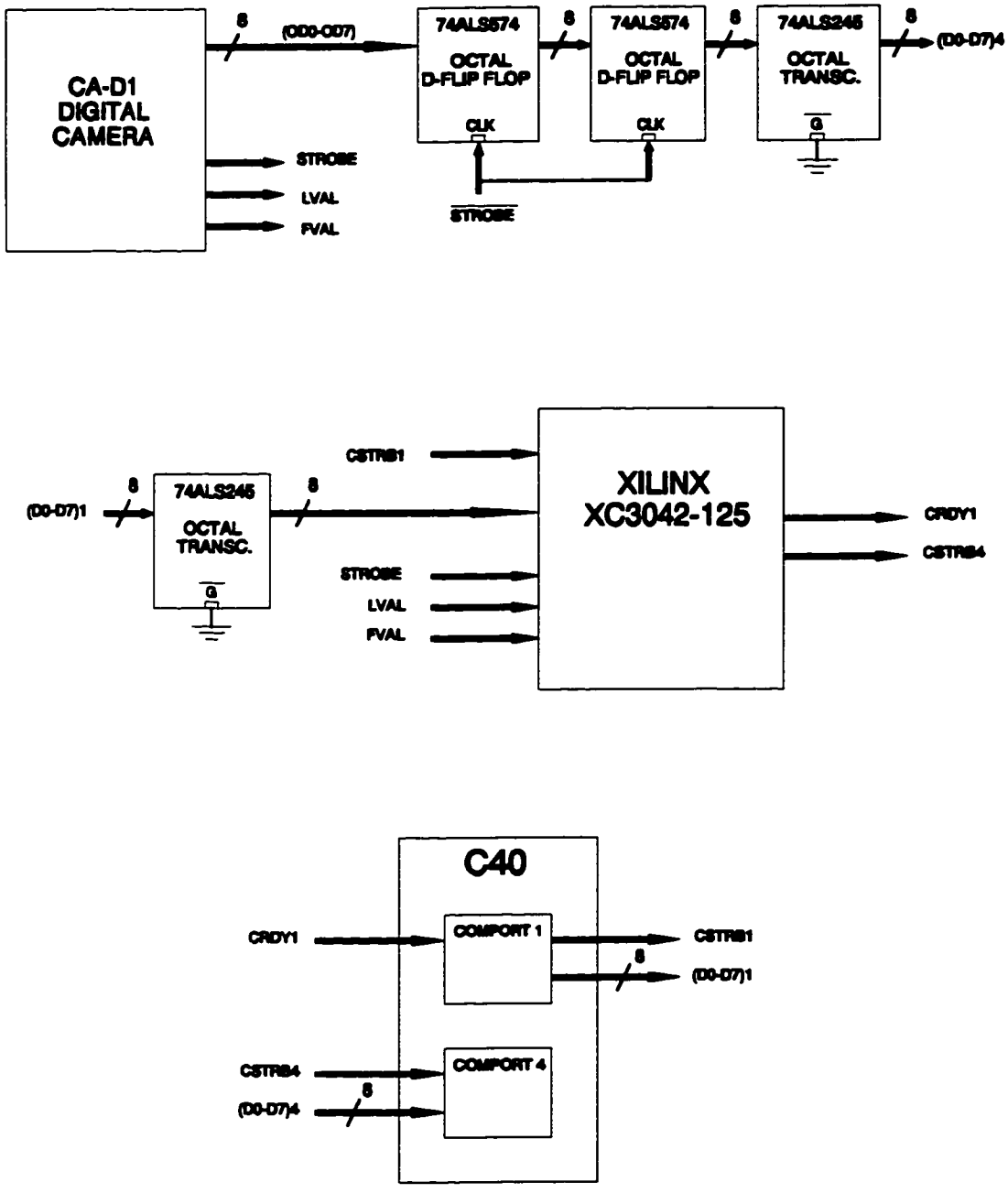


Figure 2.3. Block diagram of the window acquisition process.

(OD0-OD7)	The camera's digital output data, OD0 (LSB) to OD7 (MSB).
STROBE	The digital Pixel Valid clock.
$\overline{\text{STROBE}}$	Not Pixel Valid signal.
LVAL	Line Valid signal.
FVAL	Frame Valid signal.
$\overline{G}$	Octal transceiver enable (Low True).
CLK	Octal D-flip flop input clock.
CSTRB1	C40 communication port 1 strobe (Data Valid).
CRDY1	C40 communication port 1 ready (Data Received).
(D0-D7)1	C40 communication port 1 data bus.
CSTRB4	C40 communication port 4 strobe (Data Valid).
(D0-D7)4	C40 communication port 4 data bus.

Table 2.1. Block diagram internal signals definitions.

The C40 data strobe signal (CSTRB1), which is active low, has to be inverted to generate an active high clock. It is also fed through the Auxiliary Clock buffer (ACLK) to ensure that the C40 data input flip-flops and the shift registers are clocked synchronously. This clock (AC) is then used to load the window coordinates in the shift registers, and to start the window logic process. The Pixel Valid clock (STROBE) is inverted and fed through the Global Clock buffer (GCLK), so that the resulting clock

(PVC) can be used to synchronize all of the window logic operations. The inverter introduces a one half period delay which is necessary to synchronize the camera's Line Valid (LVAL) and Frame Valid (FVAL) signals to the PVC clock. A minor disadvantage in this synchronous window logic implementation is that the first two pixels in every line of data are always missed. Consequently, the camera's digital data has to be delayed by two pixel clocks so that the window logic applies to the correct set of pixels in every image frame. Two octal D-flip-flops are used to implement this delay (Figure 2.3), and it is important to note that an inverted Pixel Valid clock is applied to these flip-flops to match up with the inverted pixel clock PVC used in the window logic.

The pulses of the AC clock set the output of a flip-flop, which is used to set the window logic start flag (START) at the next LVAL high to low transition. The window logic enable flag (WLEN) is then set in the in-between frames time period, so that the process is always enabled before the start of the next frame. The following FVAL low to high transition sets the strobe signal enable flag STRBEN. Next, the location of the image pixels is compared with the window coordinates, and the strobe signal (CSTRB4) is activated for the pixels which are within the specified window. A Window Valid flag (XYVAL) is set to indicate that the current pixels are within the window, and CSTRB4 is used to strobe these pixels in the memory of the C40. Figure 2.4b shows the two counters and four compare circuits used to determine which pixels are within the specified image window. The X counter (XCOUNT) specifies the pixel number in a given line of data, while the Y counter (YCOUNT) specifies the line number in a given frame of data.

Two compare circuits constantly check for the condition when XCOUNT and YCOUNT equal the xmin and ymin coordinates, and the other two compare circuits check for the condition when XCOUNT and YCOUNT reach xmax and ymax. Note that the Y counter is enabled when a LVAL high to low transition occurs, so that the Y count is equal to 1 at the beginning of the first line of every frame. Therefore the range of the Y coordinates of the window has to be 1 to 256. As far as the X counter is concerned, it is enabled during the whole LVAL period, and the first pixel in the line corresponds to an X count of 0. Therefore the range of the X coordinates of the window has to be 0 to 255. Timing diagrams are included in appendix B to further describe the window acquisition process.

This DSP/FPGA combination, or windowing unit, enables the system to acquire one window of selectable size and position within the camera field-of-view, and process it. Because of the flexibility in the design, the number of windowing units in the system can easily be increased simply by adding more DSP modules to the network, together with their corresponding FPGAs to the interface board. Additional windowing units operate independently and in parallel. Currently, our experimental system uses four windowing units in order to demonstrate the advantages of this design.

Neither the processing board nor the interface board use the host computer in the operation of a windowing unit. This independence of the host computer plus the availability of the DSP's external communication ports enable the DSP to directly communicate with other systems, such as robot controllers, thus avoiding delays

associated with using the host computer's communication bus. Also, this eliminates the need to use a host computer of a certain type. For example, the network has been run using a VME bus chassis without major modifications.

2.3 WINDOWING STRATEGIES

The vision system is configurable to allow acquisition and processing of selected windows within the camera field-of-view, and the size and location of these regions are dynamically updated on a frame-by-frame basis, based on processing results such as the motion of a target. Using the high-speed communication ports of the DSPs, the windows may be made cooperative by exchanging information among processors to allow real-time adaptation to visual motion.

In biological vision, the distribution of the photoreceptors in the human eye is nonuniform with sensing elements arranged in the form of a high resolution fovea at the center of the field-of-view, surrounded by peripheral sensors with space-variant resolution. Prior work has been based on this model, such as that described in [Baron94] which uses space-variant sampling and a massively parallel SIMD computer for processing. The dynamic windowing approach we have implemented maintains the uniform image sampling using the rectangular grid of the camera, however, we collect varying resolution "Peripheral" and "Foveal" windows using the custom designed video interface board that collects pixel data from the camera for processing by the DSPs.

Based on this dynamic window design, a high speed motion analysis system has been implemented to demonstrate the use of this network in real-time applications. In general, motion analysis may be divided into three main processes [Jain81]. The first is an early detection process involving low-level processing algorithms, such as image differencing, to detect motion in the initial stage of motion analysis. The second is a peripheral process in which image data is translated into symbolic data to achieve motion tracking. The final process, or foveal process, includes the high-level processing techniques in which symbolic data is manipulated to obtain recognition results. A significant number of complex operations are required in each process, and therefore, the performance can be compromised when all three processes are run sequentially on a single processor. The architecture of the proposed vision system facilitates the implementation of the three processes in parallel through cooperative windowing schemes. Depending on the requirements of the application, at least one DSP may be assigned to perform any of the three tasks, while at the same time communicating with the other processors using the high speed communication ports of the DSP. The processors in the network can also be assigned to acquire the same window within the input image and perform various algorithms in parallel, so that the performance of the system is significantly improved. In fact, the design of this system is such that the communication delays between the different processors are minimized, the number of processors performing any one of the motion analysis tasks is not restricted, and the position updates can be directly accessed through the DSP's external communication port. Delays

associated with the use of the communication bus of the host computer are thus avoided. Multiple windows of varying sizes may be assigned as "Foveal windows" for focus-of-attention, as "Peripheral windows" for motion tracking, or as "Watch windows" for motion detection.

Prior work on multi-processor architectures includes the multiple object tracking system proposed in Fukui et al. [Fukui92] which uses a multi-window vision processor comprising 16 M68030-based processing modules and the host computer, a Sun Sparc Station. Two modules carry out the object detection, while the other 14 are reserved for object tracking, and the host computer performs the object motion interpretation. A similar configuration is described in Kubota et al. [Kubota93], with 16 M68030-based processing ("local") modules, and a Sun Sparc 2 Station as a host computer. A special hardware ("overall") unit capable of parallel and pipeline processing is also used to carry out overall image processing. Our system runs independent of the host computer, and it is flexible enough so that each DSP module may be assigned any one of the previously mentioned window functions, depending on the requirements of the application.

Our application is the real-time guidance of an industrial robot in fixtureless assembly operations. The requirements of such an application include a global fixed camera, high speed position update capability, and robust image processing techniques for initial target lock-on and recovery from loss-of-tracking. This DSP-based network may be used in applications involving robot servoing ([Allen93], [Koivo91]), or Foveal vision

using head/eye platforms as in [Murray94],[Murray93], and [Reid93]. Other applications include 3D pose determination from 2D images using inverse photogrammetry from multiple windows placed on feature points of the target, and automatic target recognition (ATR) as in [Bennam94] and [Sadjadi92]. The fixed camera scenario is one of two camera configurations which are typically used in visual servo systems: end-effector mounted, or fixed in the workspace [Hutch96]. We have implemented and experimentally verified several windowing strategies, with demonstration of tracking moving objects at frame rates of 114 frames per second. These are described in the following sections.

2.3.1 Single Processor, Multiple Function (SPMF)

A single processor-based algorithm using three modes of operation, namely "Watch" mode, "Detection" mode, and "Tracking" mode. In the Watch mode (Figure 2.4a), whole image frames, or "Watch windows" (W1), are acquired, and a motion detection algorithm is applied. The motion detection algorithm returns the coordinates of the "motion area" within the image frame, wherein motion is detected. Once motion is detected, a "Peripheral window" (W2) is defined based on the coordinates of the motion area, in order to locate the target approximately. Once the target is located, a smaller size "Foveal window" is defined about a chosen feature point of the moving object for position verification. The system then switches to the Tracking mode (Figure 2.4b). In this mode of operation, the system only acquires that section of the image frame corresponding to the Foveal window, and an area-matching algorithm is applied to update the window's

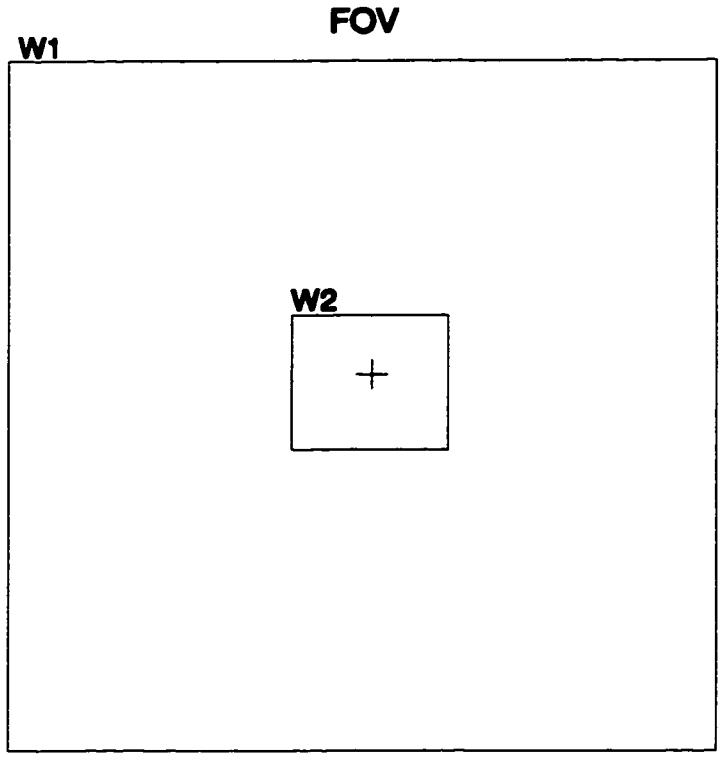


Figure 2.4a. Watch mode of SPMF.

coordinates based on the motion of the target therein (W1..W4). While moving across the field-of-view of the camera, if the object stops, or if it suddenly changes its direction of motion, the system switches to the Detection mode (Figure 2.4c), and a larger size Peripheral window is defined based on the coordinates of the last Foveal window (W1). As shown in Figure 2.4c the window size is then increased (W2,W3) until the object is once again approximately located, and the system switches back to the Tracking mode (W4). The Detection mode allows for two size increases for the Peripheral windows, at

which time the system switches back to the Watch mode, if it still fails to locate the object. A similar window placement algorithm is used in [Buttazzo94], but is software based, and therefore less likely to switch from one mode of operation to the next in consecutive frames.

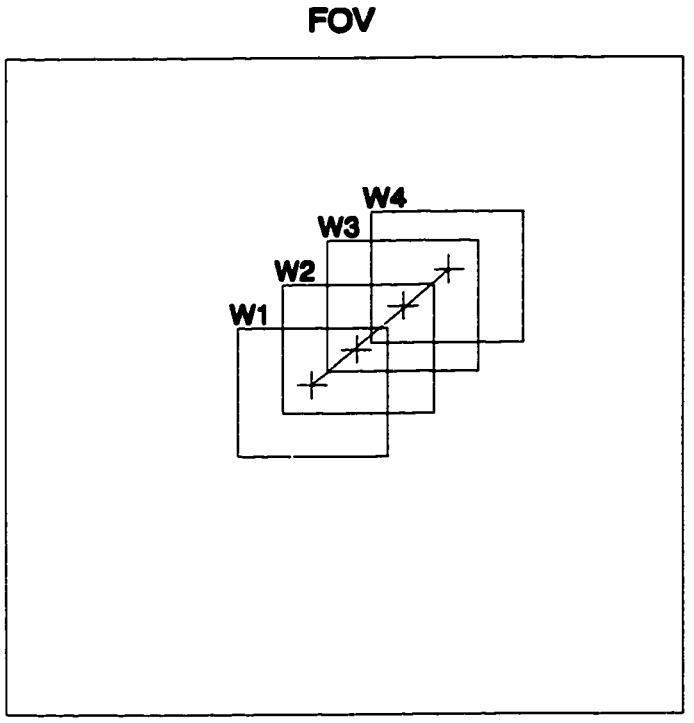


Figure 2.4b. Tracking mode of SPMF.

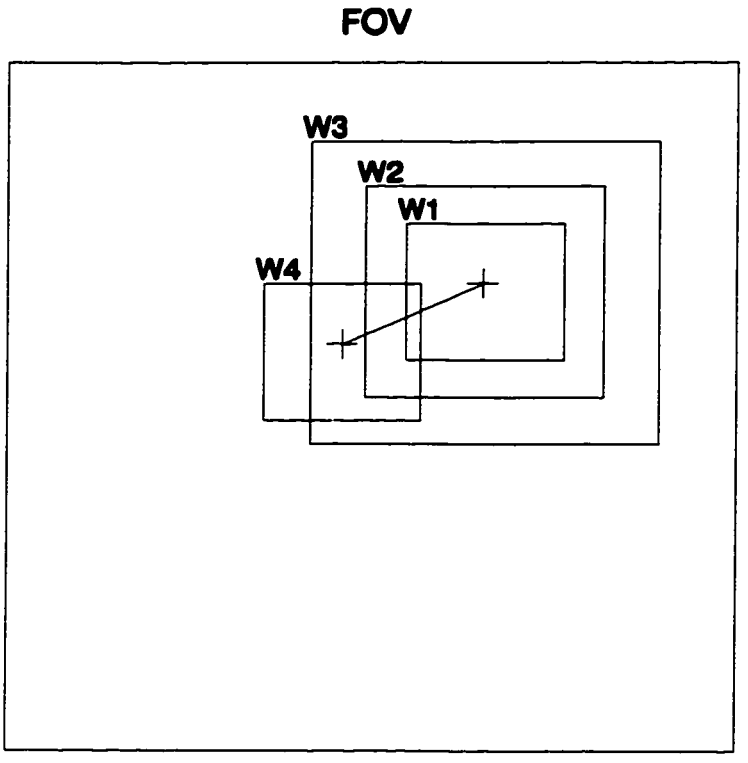


Figure 2.4c. Detetction mode of SPMF.

2.3.2 Triple Processor, Dedicated Function (TPDF)

A window placement algorithm using three processors to independently perform the "Watch", "Detection", and "Tracking" modes (Figure 2.5). In the Watch mode, whole image frames, or "Watch windows" (Ww), are acquired by the Watch processor, and a motion detection algorithm is applied. Once motion is detected (P1 to P2), the coordinates are transmitted to the Detection (Peripheral) processor, and the Peripheral window (Pw) is acquired. This processor acquires alternate pixels from every other row

of the image (25% resolution) and applies a difference measure algorithm based on cross-correlation to locate the target approximately. The coordinates of the approximate target location are then passed to the Foveal processor, which acquires the Foveal window (Fw) and applies a high resolution similarity measure algorithm to accurately determine the target position (P2). The sizes of the Peripheral and Foveal windows are chosen based on the size and speed of the moving target. Should the Foveal window lose track of the object, the Peripheral processor transmits the updated target position to the Foveal processor, and tracking is resumed. If both windows lose track of the object, the Peripheral and Foveal processors pause for the next motion coordinates to be received from the Watch processor before resuming tracking.

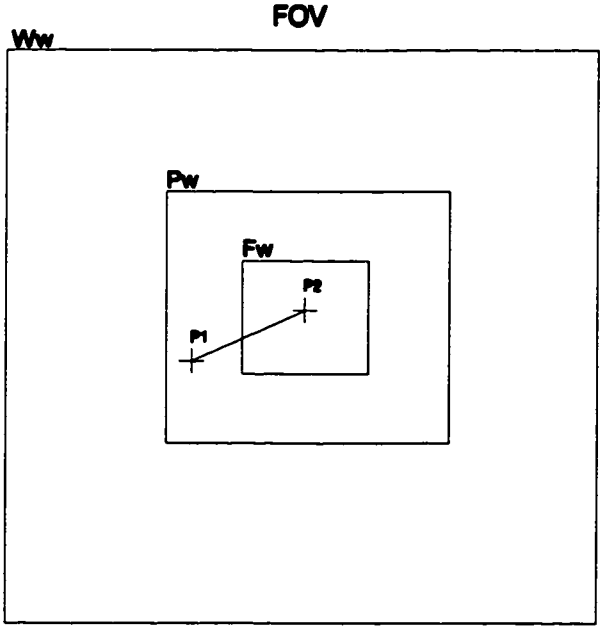


Figure 2.5. Windowing strategy of TPDF.

2.3.3 Multi-Processor, Dedicated Function (MPDF)

In this strategy, the algorithm uses multiple processors for the "Watch" mode, one for the "Detection" mode, and one for the "Tracking" mode. The Watch process of SPMF and TPDF involves the analysis of the FOV, or 64 Kb of data, which is a computational burden for one processor. However, in the Watch mode of the proposed strategy, multiple windows of different sizes and locations can be acquired depending on the application. In the example of Figure 2.6, two processors are used, so that two rectangular-shaped Watch windows (Ww1 and Ww2) can be placed along the top and bottom of the input image, where the object is expected to enter the FOV. This approach results in a quicker initial lock-on response in the Watch mode, compared with the single processor whole image frame Watch window approach of the previous two strategies. Once the object enters the FOV, the coordinates of the entry location are communicated to the Peripheral processor to begin the Detection (Pw) then Tracking (Fw) phases as described previously in strategy TPDF. In this Detection phase however, if the Peripheral window loses track of the object, the window size is increased until the object is once again located.

A summary of the window sizes used in the different modes of operation of these strategies is given in Table 2.2. These sizes have been empirically chosen to illustrate the implementation of the selected windowing strategies using this system.

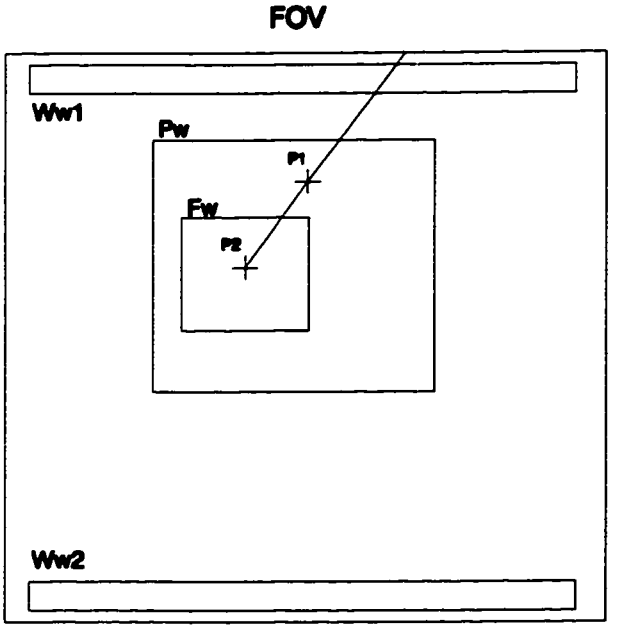


Figure 2.6. Windowing strategy of MPDF using four processors.

Window Strategies	Watch mode	Tracking mode	Detection mode
-SPMF: Watch /Peripheral / Foveal window	256x256 /Variable / 24x24	24x24	40x40 , 64x64
-TPDF: Watch Peripheral Foveal windows	256x256 Variable 24x24	No change 40x40 No change	No change 40x40 , 64x64 No change
-MPDF: Watch 1 Watch 2 Peripheral Foveal windows	256x20, Y=0-19 256x20, Y=237-256 Variable 24x24	No change No change 40x40 No change	No change No change 40x40..256x256 No change

Table 2.2. Window sizes in the different modes of operation.

2.4 Image Processing Algorithms

The motion detection method used within the Watch window is based on motion energy detection [Murray94]. By calculating the temporal derivative of the image and thresholding at a suitable level to filter out noise, we segment the image into regions of motion and of inactivity. The temporal derivative is estimated by applying simple image differencing:

$$\frac{df}{dt} \approx \frac{f(x,y,t) - f(x,y,t-\delta t)}{\delta t}$$

The target tracking algorithm applied within the Peripheral window continuously determines the approximate location of the target. A variety of low resolution template matching algorithms were evaluated for this task. In general, matching methods are classified as being either a similarity measure or a difference measure [Hussain91]. For the former, a high value and for the latter, a low value indicates a match. Given an image I(x,y) where $xmin \le x \le xmax$, $ymin \le y \le ymax$, and a template R(u,v) where: $0 \le u \le ulen$, $0 \le v \le vlen$, and $1 \le ulen \le (xmax-xmin)$, $1 \le vlen \le (ymax-ymin)$, the NCC is used to detect the instances of R(u,v) in I(x,y) as follows:

1- Normalized Cross-Correlation, or NCC (similarity measure)

$$NCC(x,y) = \frac{P(x,y)}{Q(x,y)}$$

where

$$P(x,y) \doteq \sum_{v=0}^{vlen} \sum_{u=0}^{ulen} R(u,v) \cdot I(x+u,y+v)$$

and

$$Q(x,y) \doteq \sqrt{\sum_{v=0}^{vlen} \sum_{u=0}^{ulen} R^2(u,v) \cdot \sum_{v=0}^{vlen} \sum_{u=0}^{ulen} I^2(x+u,y+v)}$$

2- Sum of Absolute Differences, or SAD (difference measure)

$$SAD(x,y) \doteq \sum_{v=0}^{vlen} \sum_{u=0}^{ulen} |R(u,v) - I(x+u,y+v)|$$

3- Sum of Squared Differences, or SSD (difference measure)

$$SSD(x,y) \doteq \sum_{v=0}^{vlen} \sum_{u=0}^{ulen} (R(u,v) - I(x+u,y+v))^2$$

The sum of squared differences weights the values, and therefore it is more sensitive to some data points being widely separated. For this reason, the SSD algorithm is chosen to be implemented within the Peripheral window. The normalized cross-correlation, which is more robust in the presence of image distortion, is considered to be the most accurate

of the three algorithms [Aschwan92]. Since the function of the Foveal window is to provide proper target identification and maintain accurate tracking of the moving object, The NCC algorithm is chosen to implement this window function based on 100% pixel resolution, in order to maintain the desired accuracy.

The computational complexity of the algorithms being applied within the different image windows varies significantly in each case. Given the input image S of dimensions NxM, the computational requirement of the detection algorithm is NxM difference operations. Therefore, the application of this algorithm within the Watch window, the size of which may be as large as the camera's FOV, will not affect the real-time performance of the system. However, the template matching methods, which involve the cross-correlation of the given template with the input image, have a computational complexity of (NxM)x((ulen+1)x(vlen+1)). In the case of the SSD algorithm, the required computational effort for every template position within the image is (ulen+1)x(vlen+1) multiplication and difference operations. Alternatively, for the NCC algorithm, the required computational effort is 4x(ulen+1)x(vlen+1) multiplications, one division and one square root operations. However, because the size of the Foveal window is small, applying the NCC algorithm within this window is not computationally demanding, and real-time performance is maintained.

2.5 Window Acquisition / Interaction

The vision system has been tested at a camera speed of 114 frames/s, which corresponds to a pixel valid (PVAL) clock rate of approximately 8 MHz. Timing diagrams for initial lock-on (Watch mode), Tracking mode, and recovery from loss-of-tracking (Detection mode) are shown in Figure 2.7. For convenient reference, key points in time (L1, L2, L3, LT, R1, R2, R3) have been indicated in this figure. The acquisition times (T_{wa} , T_{fa}) and processing times (T_{wp} , T_{fp}) of the Watch and Foveal windows respectively do not change because the window sizes remain fixed in the three modes of operation. On the other hand, the size of the Peripheral window varies from one mode of operation to the next. In fact, the window acquisition and processing times during initial lock-on (T_{pa1} , T_{pp1}), target tracking (T_{pa2} , T_{pp2}), and recovery from loss-of-tracking ((T_{pa2} , T_{pp2}) and (T_{pa3} , T_{pp3})) depend on the initial motion coordinates of the target and the Detection mode sizes as listed in Table 2.2. The time intervals of the three modes of operation are defined as follows:

Lock-on Time:

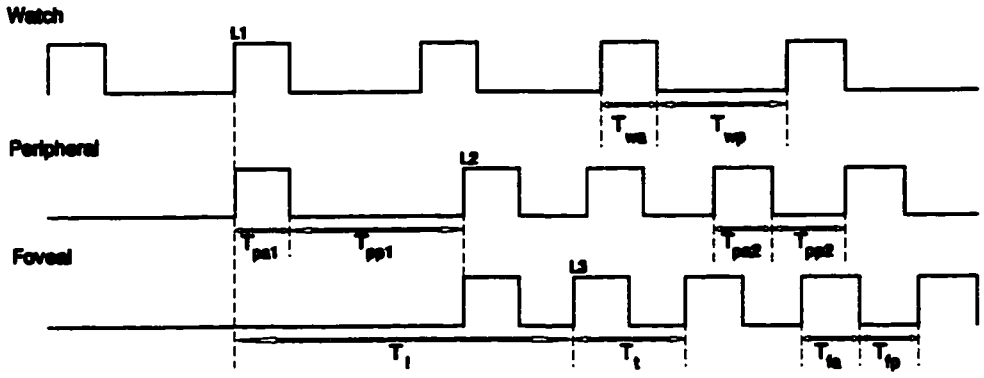
$$T_l = T_{pa1} + T_{pp1} + T_{fa} + T_{fp} \qquad (2.1)$$

Tracking Time:

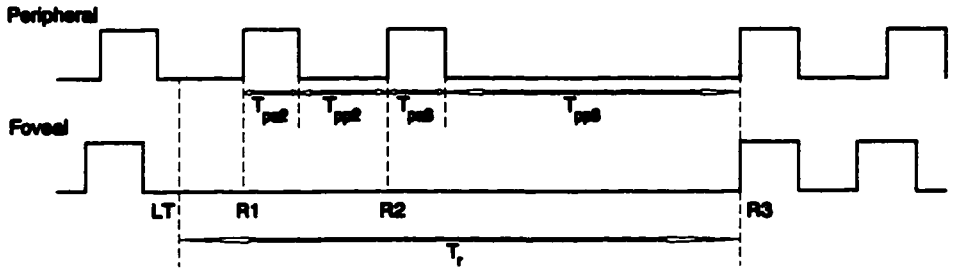
$$T_t = T_{fa} + T_{fp} \qquad (2.2)$$

Recovery Time:

$$T_r = (R1 - LT) + (T_{pa2} + T_{pp2}) + (T_{pa3} + T_{pp3}) \qquad (2.3)$$



(a)



(b)

Figure 2.7. Timing diagrams of the three modes of operation: **a)** Initial lock-on and target tracking. **b)** Recovery from loss of tracking.

The sequence of operations in each of the three windowing strategies to initially lock on the target begins in the Watch mode, with consecutive Watch windows being sampled to detect motion. Once motion is detected (L1), the position and size of the initial Peripheral window is defined based on the motion coordinates received from the Watch window. The sizes of subsequent Peripheral windows are based on the Detection mode of the three strategies listed in Table 2.2. In the Peripheral window, the Detection algorithm is applied to determine the approximate location of the moving target, and the window's position is updated accordingly. Once the approximate target location is determined (L2), it is passed on to the Foveal window to verify the target's location (L3), which completes the interval for initial lock-on (Eq.(2.1)). However, in strategies TPDF and MPDF, the Watch, Peripheral, and Foveal windows continue to be acquired in parallel after initial target lock-on, which ensures a quicker response to target loss of tracking. Next, the Tracking mode is activated, and the Foveal window maintains accurate tracking of the target's motion. The target tracking time equals the acquisition plus the processing times of the Foveal window (Eq.(2.2)). If the Foveal window loses track of the target (LT), the Detection mode is activated, and the Foveal processor stands by until it receives the approximate target location from the Peripheral processor. If the target is not located at the end of the processing time of the current Peripheral window (R1), the position of the next Peripheral window is adjusted based on the coordinates of the last Foveal window before attempting to relocate the target (R2). If the attempt is not successful, the size of the following Peripheral window is increased to improve the chances of relocating the target (R3). However, if the second attempt also fails, the

Watch mode is activated, and the sequence of operations for initial target lock-on is restarted. If the current Peripheral window locates the target, the recovery time is (R1 - LT), whereas if the target is located at R2, the recovery time increases to: (R1 - LT) + (T_{pa2} + T_{pp2}). In the worst case, the target is located at R3 and the recovery time is given by Equation (2.3).

A summary of typical measured times for lock-on, tracking, and recovery from loss-of-tracking for each of our three strategies is given in Table 2.3.

Window Strategies	T_l (ms)	T_t (ms)	T_r (ms)
-SPMF:	320	18	112
-TPDF:	230	18	97
-MPDF:	80	18	97

Table 2.3. Average lock-on, tracking, and recovery times in the different modes of operation.

2.6 Experiments

In order to demonstrate the performance of this vision system in tracking moving objects, the experimental setup involved planar motion tracking of a target which was attached to the end-effector of a robot arm. The CA-D1 camera, which was operating at a frame rate of 114 frames per second, or 8.8 ms per frame, was mounted at a fixed height of 800 mm, and had a focal length of 25mm. The robot, which was a five degrees of freedom CRS Robotics A255 arm, was programmed to move the object in an XY path (Figure 2.8) at speeds of up to 60 mm/s. This resulted in an equivalent velocity of approximately 100 pixels/second. The resulting motion of the object was similar to that of a pendulum, moving in and out of the FOV of the camera in a continuous manner.

The tracking algorithm made no assumptions about the motion of the object, i.e. no *a priori* knowledge of the object's path was needed to maintain tracking. The algorithm was set to track the center point of the object, so the Foveal window would be centered about the selected point. The template size used in the matching algorithm was chosen to be 20x20, and the Foveal and Peripheral window sizes were chosen to be 24x24 and 40x40 respectively.

The experimental results clearly demonstrated the advantage of using separate processors for the "Watch", "Detection", and "Tracking" modes over the single processor approach, especially when the moving object undergoes a sudden change in direction.

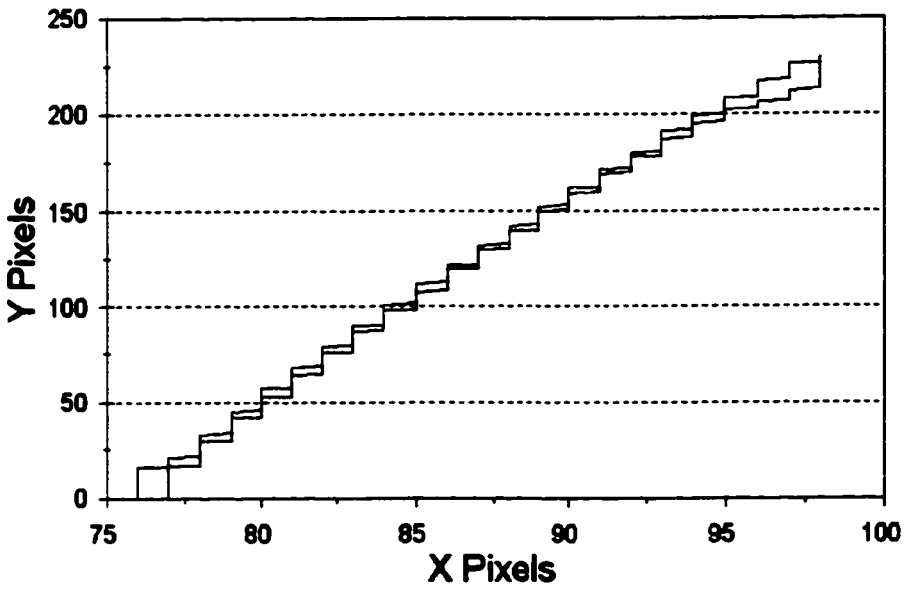


Figure 2.8. The actual XY path taken by the moving object.

Such a scenario forced the single processor to switch to several windows of increasing sizes before the target could be relocated, which results in a time delay of several image frames and an increase in the RMS error. This is more obvious at higher speeds , and the maximum tracking speed of this strategy was found to be 80 pixels/second beyond which tracking was completely lost (Figure 2.9). In contrast, for the multi-processor case, the Foveal window could be redirected in the next image frame, based on the coordinates received from the Peripheral processor. This resulted in a RMS pixel error of around

0.25 pixels for speeds of up to 80 pixels/second, and a maximum tracking speed of 100 pixels/second (Figure 2.9).

The use of two rectangular-shaped Watch windows in strategy MPDF had a significant effect compared with the one window whole image frame approach of strategy TPDF. The coordinates of the motion area were determined and transmitted to the Peripheral window at a higher rate, which resulted in a faster target lock-on time than that of strategy TPDF (Figure 2.10). However, the limited function of these Watch windows in determining the motion area, because of their smaller sizes, caused a larger RMS position tracking error at higher target speeds (Figure 2.9).

Based on a template size of 20x20, the rate of the Foveal processor is 18 ms, or half the frame rate of the camera, in 70% of the image frame. In the bottom 76 lines of the pixel array, the rate decreased to 27 ms due to the processing delays which caused the following image frame to be missed. Smaller template sizes such as 16x16 were also tested, and the rate was measured at 8.8 ms. Using template sizes of 16x16 or smaller, the target tracking could be performed on a frame-by-frame basis, and the target position updates could be provided every 8.8 ms. These position updates could also be communicated to other processors directly using the DSP's high speed communication ports.

Our current experimental setup includes a 5-axis CRS Robotics A255 robot arm

which uses a transputer-based controller. We have added a transputer link interface module to the processing board in our system, so that the DSP output is simply redirected to the robot controller for motion control applications. The vision system can provide the robot controller with target position updates at high rates, as high as 114 Hz in certain cases. This allows real-time visual path control updates, since the motion control update rate of the robot is typically 10 ms.

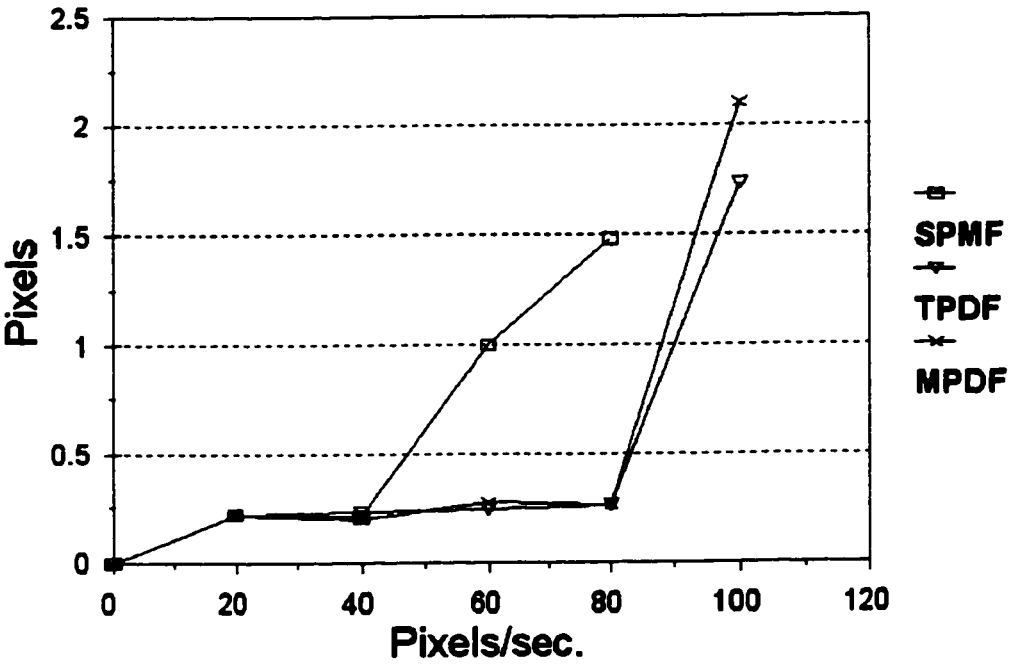


Figure 2.9. RMS position error of the Foveal window centroid based on the three strategies versus target speed.

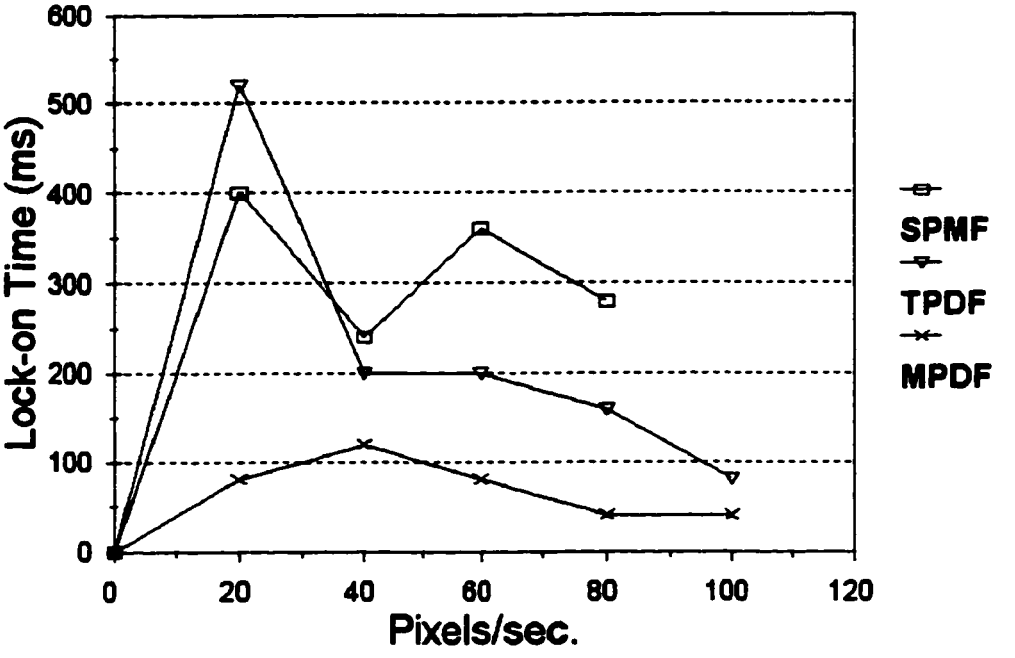


Figure 2.10. Time required by the three strategies to initially lock on the target versus target speed.

CHAPTER 3

ROBUST REAL-TIME CORNER LOCATION MEASUREMENT

3.1 Introduction

Corners are useful features to be extracted from images because they are invariant to image translation, rotation, and change of size. Model-based approaches such as template matching are sensitive to image rotation and change of size, and their use would require a library of all possible model orientations and sizes, which is not practical. Hence, corner detectors (CD) are generally preferred to model-based algorithms in computer vision tasks, such as solving the image correspondence problem. For example, in Huertas [Huertas81] corners are used to detect buildings in aerial images, and in Frendo [Frendo89], corners are used to track 4-point planar patterns in 3D. In general, an image feature is called a corner where two edge boundaries meet, or where the direction of the edge boundary is changing rapidly. The different approaches to corner detection may be divided into two categories:

1- Methods which rely on prior segmentation of the image and subsequent analysis of region boundaries.

2- Methods which operate directly on the gray scale image.

An example of the first class of methods is that of Jain *et al.* [Jain95b] which uses a corner detector that requires fitting lines to edge points, and then computing the intersection of the lines. Such a method is clearly dependent on the success or failure of fitting lines to edge points, and this can have a negative effect on the overall performance of the corner detector. For example, considering a list of edges from two adjacent sides of a rectangular-shaped object, the likelihood of assigning some edge points to the wrong side is significant in the neighborhood of the corner point. Therefore, the lines may not be fitted to the edges properly, which consequently results in an error in the corner location measurement.

Examples of the second category include the works of Haralick and Shapiro [Haralick93], Wang and Brady [Wang94], and Gaiarsa and Capson [Gaiarsa94]. In Haralick and Shapiro [Haralick93], several edge detection methods which operate directly on the gray scale image are presented. They include computing the incremental change in gradient direction along the tangent line to the edge at the point that is a corner candidate, or evaluating the incremental change along the contour line that passes through the corner candidate. The main advantage of such corner detectors is that their performance is not dependent on the success or failure of a prior segmentation step as in [Jain95b], however, the computational expense is more significant. Also, in real life images which include scattered edge points, edge points fitting the characteristics of the candidate corner point may be more than one, which increases the chances of misclassification.

The corner detection algorithm suggested by Wang and Brady [Wang94], is based on the observation of surface curvature. A measure of corner detection consistency is combined with a measure of accuracy in corner localisation to achieve optimal trade-off. Since surface curvature is proportional to the second derivative of the unit tangent vector along the edges, a smoothing operation using Gaussian convolution is applied to reduce the effects of noise. However, this operation causes a displacement in the corner location measurement, proportional to the standard deviation of the Gaussian convolution. Therefore, additional constraints on the equations of the algorithm have to be introduced, which is bound to restrict the performance of the corner detector. Also, the authors do not provide a quantitative measure to demonstrate the advantages of the proposed approach.

In Gaiarsa and Capson [Gaiarsa94], the proposed corner detector determines the object corner location based on the area and the XY moments of the portion of the planar shape contained within the image window. In addition to this information, the geometric properties of each intersection case of the shape with the window sides are used to locate the corner. The intersection case is determined based on the number of intersections with the sides of the window. However, this technique is not evaluated in situations where the sides of the object are made up of jagged edges, which is often the case in real applications.

The corner detection algorithm in this paper exploits the geometry of the object

within the image window, without having to compute the object area and XY moments as in [Gaiarsa94]. Furthermore, it has the advantage that it performs robustly in the presence of pixel intensity variations (noise) and a non-uniform background which makes target tracking harder (clutter). The edge points are first determined, and a clutter removal algorithm using pixel labelling and morphological filtering is applied next. The bounding box of the filtered edge pixels is then extracted, and the corner location is measured based on the relationship of the edge pixel vertices on the sides of the bounding box. A search algorithm is also used to efficiently determine the corner location with a minimum of distance computations. An information content measure based on the number of edge points in the image is developed and applied to determine how hard it is to detect and track the corner point. The peak value corresponds to the number of edge pixels required to yield the lowest tracking error. The metric is also compared to other measures [Bhanu86] to demonstrate its advantages.

The performance of the corner detection is demonstrated using a real-time motion tracking experiment on a precision motion table with an accuracy of 0.005 mm. The computed results were compared to the coordinates obtained from the motion table. It was found that an accuracy of 0.5 pixel RMS (0.25 mm) could be obtained at 114 Hz.

3.2 Corner Detector

The proposed corner detection algorithm exploits the geometry of the object within the image window. A Sobel filter [Sobel70] is initially used to detect the edges within the input gray scale image, and hence generate the corresponding binary image (edge map). A pixel labelling algorithm for clutter removal is applied next, followed by morphological filtering. The corner detection algorithm is then used on the filtered image for accurate measurement of the location of the corner. A data flow diagram of this corner detection strategy is given in Figure 3.1.

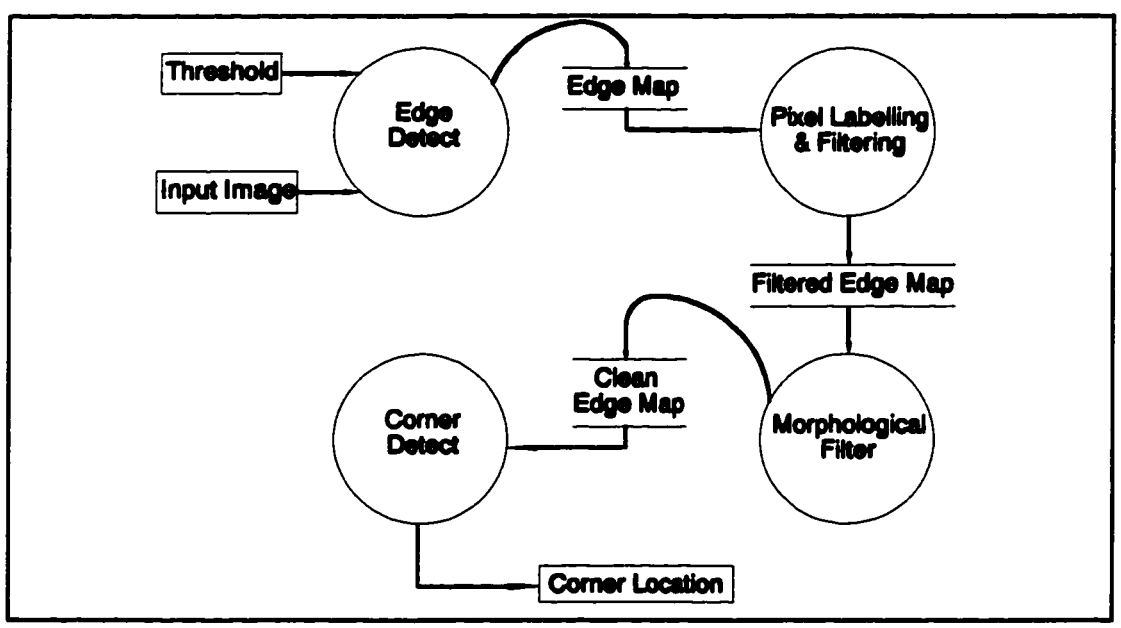


Figure 3.1. Data flow diagram of the corner detection strategy.

3.2.1 Pixel Labelling

Once the edge map is generated, a pixel labelling algorithm is applied to remove clutter from the image. Given the x and y coordinates of an edge pixel $P_k[x_k][y_k]$ to be x_k and y_k, $P_i[x_i][y_i]$ and $P_j[x_j][y_j]$ are defined to be neighbors :

if $(| x_i - x_j | \leq 1) \bigwedge (| y_i - y_j | \leq 1)$.

For a given set of labels L_p ($p = 1, 2, 3, ..$), if P_i and P_j are neighbors, then

$label(P_i) = label(P_j)$.

The edge pixels with the label L_t , which has been assigned the most number of times, are preserved, and all other edge pixels are deleted from the edge map.

3.2.2 Morphological Filtering

The proposed pixel labelling algorithm is useful for removing patches of clutter which are away from the object. However, in cases where these patches are neighboring the object, the algorithm will add them to the original shape of the object instead. Such cases, require mathematical morphology to help distinguish the object boundary from clutter. In our application a morphological "Opening" [Jain95a] is applied, which involves a binary *erosion* followed by a binary *dilation*. Both the *erosion* and *dilation* operations are performed with a "Plus-shaped" structuring element. This operation filters the edge map further and yields the clean edge map.

3.2.3 Corner Detection

Given the clean edge map of the object within the Foveal window, the bounding box of the edge pixels is extracted. The vertices are then determined, and the corner location is measured based on the relationship of these vertices. A total of eight points are needed to determine the bounding box and the corner location. These points define the minimum (Min) and maximum (Max) edge pixel location in the image I(x,y) both in the X and Y directions.

-First Minimum Bounding X :

$$P_{00}[x_{00}][y_{00}], \qquad x_{00}=Min_x(x,y)\in I, \qquad y_{00}=Min_y(x_{00},y)\in I$$

-Last Minimum Bounding X :

$$P_{01}[x_{00}][y_{01}], \qquad y_{01}=Max_y(x_{00},y)\in I$$

-First Maximum Bounding X :

$$P_{10}[x_{10}][y_{10}], \qquad x_{10}=Max_x(x,y)\in I, \qquad y_{10}=Min_y(x_{10},y)\in I$$

-Last Maximum Bounding X :

$$P_{11}[x_{10}][y_{11}], \qquad y_{11}=Max_y(x_{10},y)\in I$$

-First Minimum Bounding Y :

$$\overline{P_{00}}[\overline{x_{00}}][\overline{y_{00}}], \qquad \overline{x_{00}}=Min_x(x,\overline{y_{00}})\in I, \qquad y_{00}=Min_y(x,y)\in I$$

-Last Minimum Bounding Y :

$$\overline{P_{01}}[\overline{x_{01}}][\overline{y_{00}}], \qquad \overline{x_{01}}=Max_x(x,\overline{y_{00}})\in I$$

-First Maximum Bounding Y :

$$\overline{P_{10}}[\overline{x_{10}}][\overline{y_{10}}], \qquad \overline{x_{10}}=Min_x(x,\overline{y_{10}})\in I, \qquad \overline{y_{10}}=Max_y(x,y)\in I$$

-Last Maximum Bounding Y :

$$\overline{P_{11}}[\overline{x_{11}}][\overline{y_{10}}], \qquad \overline{x_{11}}=Max_x(x,\overline{y_{10}})\in I$$

Next, the required vertices to locate the corner are defined based on the coordinates of the bounding box. Typically, the bounding box is defined by three of the eight minimum and maximum edge point locations (Figure 3.2a), and by choosing the same two points, Q_1 and Q_2 , out of the total eight, the third point (Q_3) is selected from the remaining six based on the corner location (Figure3.2b).

Q_1 is chosen to be the first minimum bounding X :

$$Q_1 = P_{00}$$

Q_2 is chosen to be the first maximum bounding X :

$$Q_2 = P_{10}$$

(Q_2 is distinct unless the object is a vertical line)

The selection of Q_3 is such that it remains distinct from Q_1 and Q_2 :

$$Q_3 = \begin{cases} \overline{P_{00}}, \text{ if } (\overline{P_{00}} \neq Q_1) \wedge (\overline{P_{00}} \neq Q_2) \\ \overline{P_{10}}, \text{ if } (\overline{P_{10}} \neq Q_1) \wedge (\overline{P_{10}} \neq Q_2) \\ \overline{P_{01}}, \text{ if } (\overline{P_{01}} \neq \overline{P_{00}}) \\ \overline{P_{11}}, \text{ if } (\overline{P_{11}} \neq \overline{P_{10}}) \\ P_{01}, \text{ if } (P_{01} \neq P_{00}) \\ P_{11}, \text{ if } (P_{11} \neq P_{10}) \end{cases}$$

else Q_3 is undefined.

The midpoints between Q_1 and Q_2 (M_{12}) and Q_1 and Q_3 (M_{13}) are computed, and the number of edge pixels in a 3x3 neighborhood (MidNum) about each one of these midpoints is determined (Figure 3.2b). The corner location logic is given as follows :

if Q_3 is defined then

$$\text{-The corner is } \begin{cases} Q_3 & \text{if } (MidNum(M_{12}) = 0) \\ Q_2 & \text{if } (MidNum(M_{12}) > 0) \bigwedge (MidNum(M_{13}) = 0) \\ Q_1 & \text{if } (MidNum(M_{12}) > 0) \bigwedge (MidNum(M_{13}) > 0) \end{cases}$$

else if Q_3 is undefined (The bounding box is defined by two vertices only, Figure 3.2c)

-Use the method based on the Golden Section Search technique.

3.3 Golden Section Search

The use of the above-mentioned Boolean logic for corner detection is not possible in the case of an undefined vertex Q_3. An alternative approach is adopted instead, whereby the distance D_p from the edge pixels to the line joining the two vertices Q_1 and Q_2 is computed, and the corner is determined by the edge pixel yielding the maximum distance. The computational complexity of this method is significantly reduced by using a maximization of function method such as the Golden Section Search.

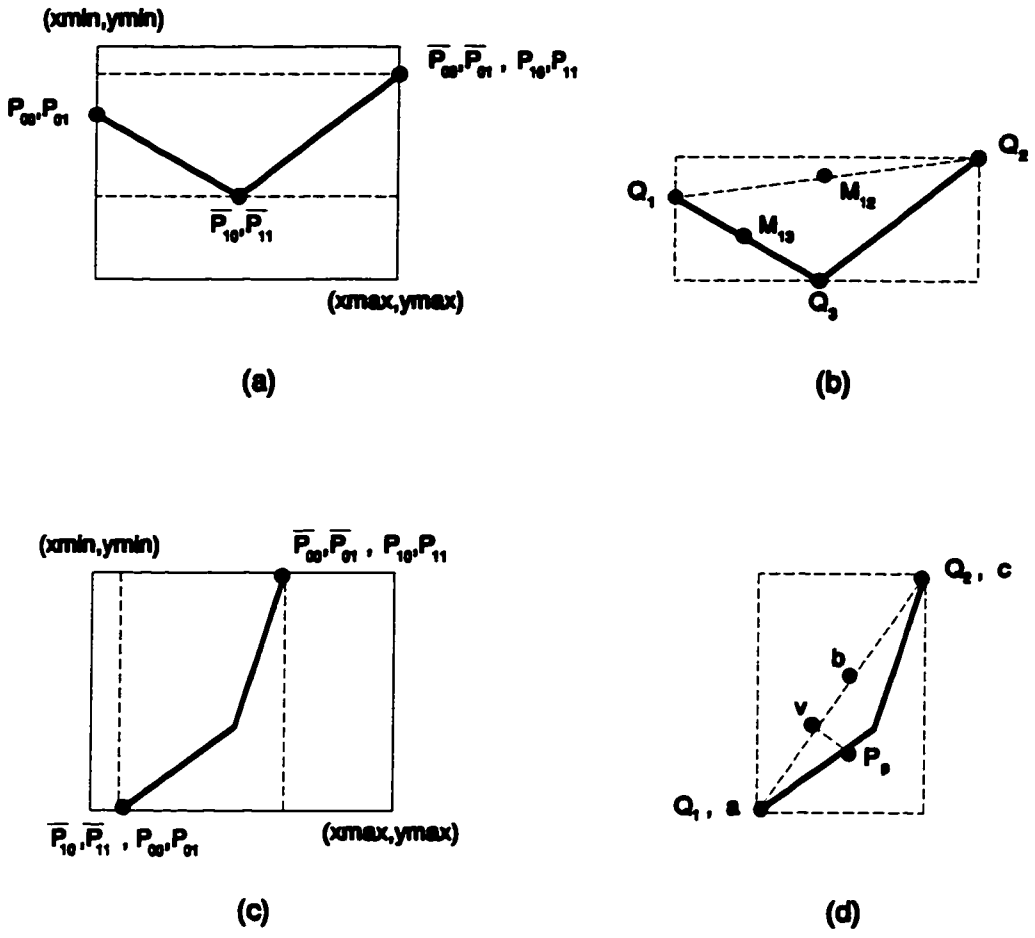


Figure 3.2. Bounding box extraction and corner detection when Q_3 is defined ((a), (b)), and when Q_3 is undefined ((c),(d)).

Given a single function F(x) ($0 \leq x \leq 255$), the Golden Section Search (GSS) method determines the value of x where F(x) takes on a maximum value, and calculates the value of F(x) which is achieved at the maximum. Since the cost of the evaluation of the function is the dominant computational effort, these steps are performed while F(x) is evaluated as few times as possible.

A maximum of a function is known to be bracketed only when there is a triplet of points $a < b < c$, such that F(b) is greater than both F(a) and F(c). In this case the function is known to have a maximum in the interval (a,c) (if it is non-singular). A new point v is chosen next, either between a and b or between b and c. As an example, suppose that the latter choice is made, and F(v) is evaluated. If $F(b) > F(v)$, then the new bracketing triplet of points is $a < b < v$, otherwise, if $F(b) < F(v)$, then the new bracketing triplet is $b < v < c$. In all cases the middle point of the new triplet is the abscissa whose ordinate is the best maximum achieved so far. The process of bracketing is continued until the distance between the last two outer points of the triplet (b1 and b2) is sufficiently small, i.e. within a pre-defined tolerance "*tol*".

The optimal bracketing interval $a < b < c$ has its middle point b a fractional distance 0.38 from one end (such as a), and 0.62 from the other end (such as c) [Press90]. The number of function evaluations required by the Golden Section Search method is proportional to :

$$\log_2 \frac{(b1-b2)}{tol},$$

Given $Q_1[x_{00}][y_{00}]$ and $Q_2[x_{10}][y_{10}]$, the slope of the equation of the line L joining the two points is computed :

$$m = \frac{y_{10}-y_{00}}{x_{10}-x_{00}}$$

If the line L is vertical ($x_{00} = x_{10}$), or horizontal ($y_{00} = y_{10}$), then there is no corner to locate and the algorithm is stopped. Otherwise, the algorithm is continued to determine the maximum distance between the edge pixels and L .

The distance D_L between Q1 and Q2 has to be computed before the initial bracketing triplet are determined :

$$a = x_{00}, \qquad c = x_{10}, \qquad \text{and} \qquad b = 0.62*D_L .$$

The new bracketing points coordinates are given by $v(x_v, y_v)$, where :

$$x_v = x_{00} + \Delta_v, \qquad y_v = y_{00} + m*\Delta_v,$$

and Δ_v is given by the iterative results of the Golden Section Search. Based on every bracketing point v, an iterative process is initiated along the line L_p that is perpendicular to L to locate the last edge pixel (P_p) on L_p, and then compute the distance D_p from P_p to v (Figure 3.2d). This distance represents the function value based on v (F(v)), which

is returned to the GSS algorithm to determine whether it is the maximum of the function or not.

The iterative process along L_p begins by computing the incremental change Δ_p , where : Δ_p = Minimum(|m| ,1).

Given the product of the slopes of the two perpendicular lines (L and L_p) :

$$m*m_p = -1 ,$$

$P_p[x_p][y_p]$ is determined :

$$x_p = x_v \pm \Delta_p * i , \qquad (3.1a)$$

$$y_p = y_v \pm m_p * \Delta_p * i , \qquad (3.1b)$$

and i is the iteration variable. For a given row of the edge map, corresponding to y_v, if the X coordinates of the edge pixels are greater than x_v, the positive signs are used in equations (3.1), otherwise the negative signs are used. The iterations continue until P_p is found to be the last edge pixel along L_p . Once P_p is determined, the distance from P_p to v is computed :

$$D_p = \sqrt{(x_p - x_v)^2 + (y_p - y_v)^2}$$

This value of D_p is then returned to the GSS algorithm, and the search continues until the maximum value is found.

3.4 Experimental Results

To demonstrate the performance of this corner detector, the experimental setup involved planar motion tracking of a metal part on the surface of an XY table in the presence of clutter (Figure 3.3). The digital camera, which was operating at a frame rate of 114 frames per second, or approximately 9 ms per frame, was mounted at a fixed height of 800 mm, and had a focal length of 25 mm. The XY table, based on the Techno Isel C-Series controller, was programmed to move the object in a triangular XY path (Figure 3.4).

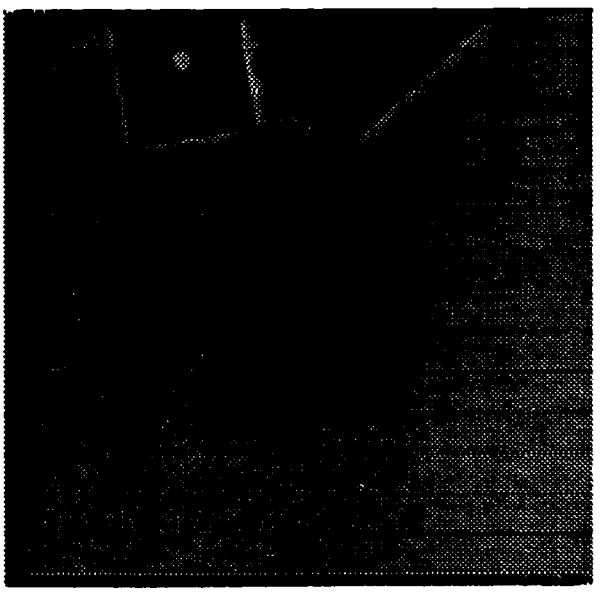


Figure 3.3. The metal part (left) used in the experimental setup.

The tracking algorithm made no assumptions about the motion of the object, i.e. no *a priori* knowledge of the object's path was needed to maintain tracking. The algorithm was set to track the corner point of the object, so the Foveal window would be centered about the selected point. The template size used in the matching algorithm within the Peripheral window was chosen to be 16x16, and the window sizes were chosen to be 20x20 and 40x40 for the Foveal and Peripheral windows respectively. Based on these window sizes, the corner location was measured on a frame-by-frame basis, or 114 Hz.

Because the performance of the corner detector is based on the resulting clean edge map, the evaluation process was performed over a wide range of edge detection thresholds (TH). The experimental results demonstrated that for an average image gray scale intensity of 80, a mean error of around 0.3 pixels (Figure 3.5) and an RMS error of around 1 pixel could be maintained within a threshold range $30 < TH < 70$, both in the X and Y directions (Figure 3.6). However, the corner detection performance deteriorated using threshold values outside this range.

The experiments also included the computation of the rate of loss-of-tracking (RLT) over the entire path based on the different TH values. The advantage of using the proposed multi-windowing strategy over a single window approach was demonstrated by comparing the RLT of both approaches (Figure 3.7).

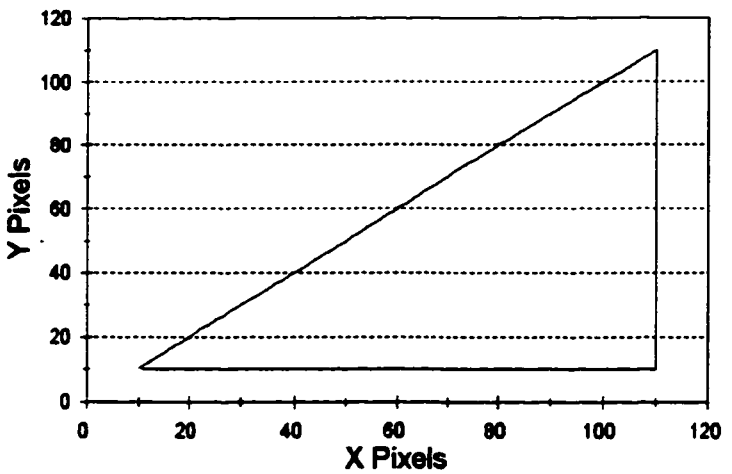


Figure 3.4. Actual XY path taken by the moving object.

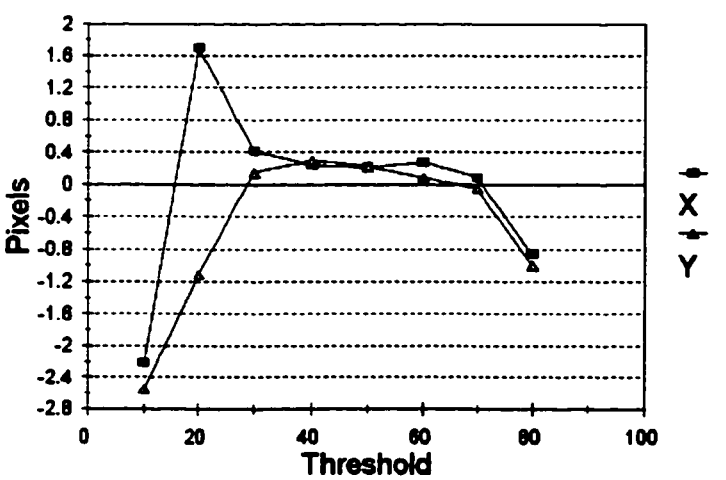


Figure 3.5. Mean error of the corner location measurement in the X and Y directions.

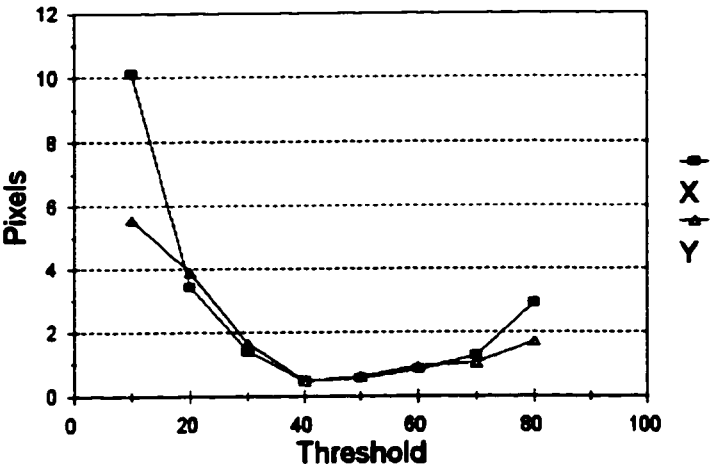


Figure 3.6. RMS error of the corner location measurement in the X and Y directions.

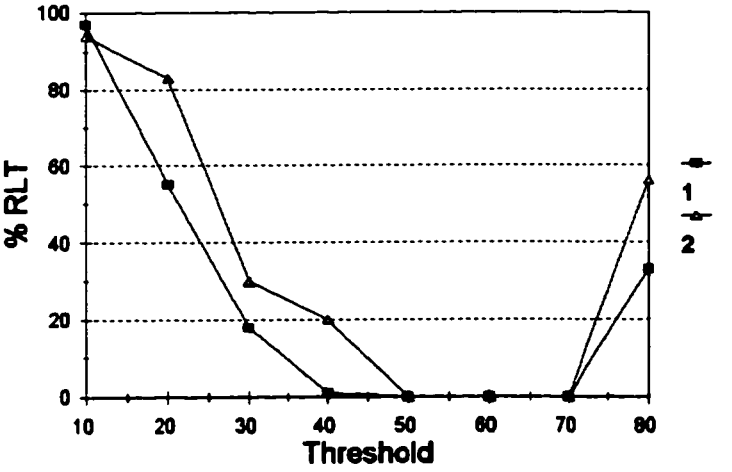


Figure 3.7. RLT percentage using the multi-windowing strategy (1), and a single window approach(2).

3.4.1 Information Content Measure for Corner Detectors

The evaluation of the corner detection performance based on the edge pixels information content ([Bhanu86]) within the Foveal window was performed. This information content is measured by finding the pixels in the image at which the magnitude of the edge detection operator exceeds TH. Then, an information content measure I is defined by :

$$I = -\text{Log}_2 P, \qquad (3.2)$$

where P is the probability of possible pictures made up of edge pixels. This edge pixel information content measure was evaluated based on different values of TH (Figure 3.8). Then, by plotting the RMS error versus the information content (Figure 3.9), the optimum point in the plot was found to correspond to a threshold value within the range of 30 to 70.

This information content measure (Eq.(3.2)) is expected to yield a high value for a large number of edge pixels, and a low value for a small number of edge pixels. The same is true for similar measures listed in [Peters88]. However, the corner detection performance measure should peak when the number of edge pixels within the image window represents a clear set of edges, which may not be possible when the number of edge pixels is maximum. Whether the window is full of edge pixels, or contains a minimum number, the information content measure should be at its minimum because the corner location error will be maximum.

We propose computing the inverse of the corner location error (CLERR) based on the number of edge pixels as a new information content measure (I_{cd}) to evaluate the performance of the corner detector :

$$I_{cd} = \frac{1}{CLERR}$$

Plotting I_{cd} (Figure 3.10) shows that the maximum value of this information content measure corresponds to the number of edge pixels that yields the minimum corner location error, while the maximum and minimum points of Figure 3.8 correspond to the thresholds that yield the maximum errors.

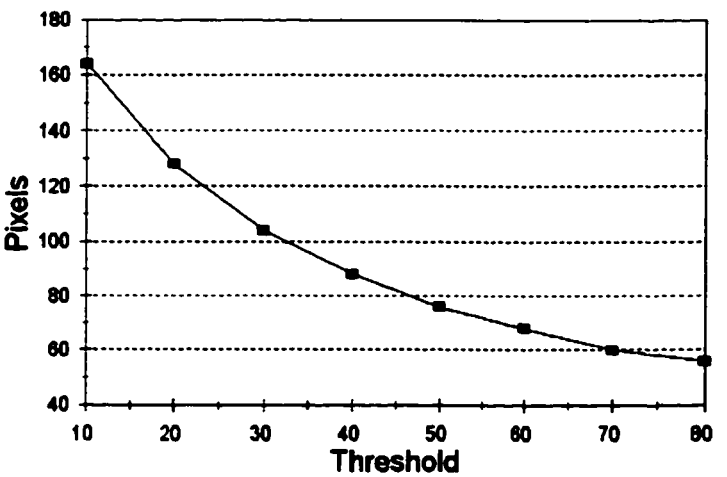


Figure 3.8. Edge pixel information content in the image based on the edge detect threshold selection

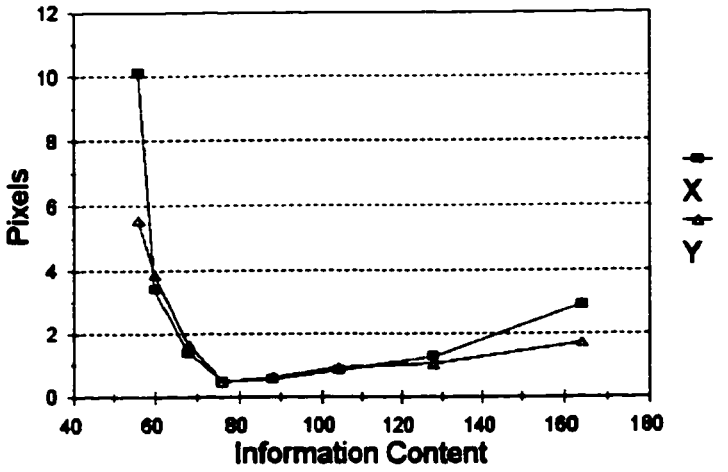


Figure 3.9. RMS error in the X and Y directions based on the edge pixel information content in the image

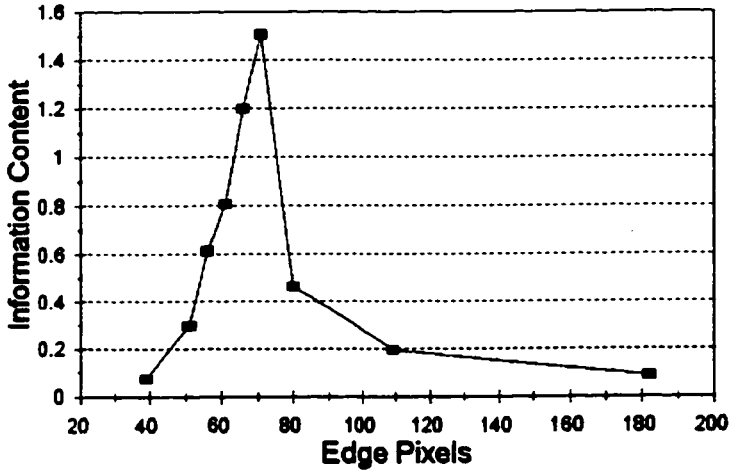


Figure 3.10. Information content measure for corner detection evaluation.

CHAPTER 4

COOPERATIVE WINDOWING

4.1 Introduction

The results of using the windowing strategies and the image processing algorithms of chapters three and four prompted the development of a 'cooperative' windowing scheme to improve the tracking performance of the system.

Using windowing strategy TPDF, which implements the three processes of motion tracking in parallel, both a correspondence-based (NCC) and a feature-based method (CD) were applied within the Foveal window to achieve real-time tracking. However, the NCC approach relies on gray value math, which makes it sensitive to changes in illumination and occlusions. Also, because the corner detector depends on edge detection, it is susceptible to mistracking due to background or foreground occluding edges and clutter [Hager96]. Since the disadvantages of both methods manifest themselves in opposite scenarios, the two approaches are integrated in two separate but networked Foveal windows to improve the tracking performance.

4.2 Cooperative Windowing Strategy

The cooperative windowing scheme (COOP) involves one watch window, one peripheral window, and two networked Foveal windows (Fw1 and Fw2) (Figure 4.1). The Foveal windows processors, DSP2 and DSP3 (Figure 2.1), maintain a constant communication link to ensure that Fw1 and Fw2 are acquired at the same time based on the same window coordinates. The tracking results of both windows are compared and weighed using a measure of confidence, and the window coordinates are updated accordingly.

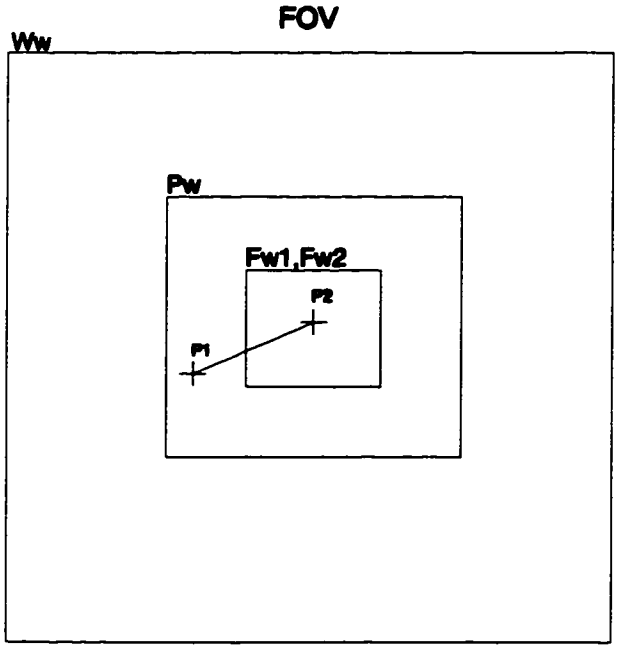


Figure 4.1. Cooperative windowing strategy.

4.3 Window Confidence Measure

The applied window confidence measure is designed to determine the variations in the results of each algorithm (NCC and CD). The algorithm that computes data with lower variation is selected, and the coordinates of the Foveal windows are updated based on these results.

The adopted method consists of finding a least squares line for the NCC and the CD data sets and comparing their respective sum of squares of the errors (SSE) before a decision is made. Given a data set NCC(p,t) of size n, which represents the Foveal window centroid position (p_i) over time as computed by NCC, where p_i is determined by its X and Y coordinates (x_i and y_i) :

$$p_i = \sqrt{x_i^2 + y_i^2}$$

the least squares line equation is defined to be:

$$\hat{p} = at + b$$

where

$$a = \frac{SS_{pt}}{SS_{tt}}$$

and

$$SS_{pt} = \sum t_i p_i - \frac{\sum t_i \sum p_i}{n}$$

and

$$SS_{tt} = \sum t_i^2 - \frac{(\sum t_i)^2}{n}$$

and

$$b = \frac{\sum p_i}{n} - \frac{a \sum t_i}{n}$$

The sum of squares of the errors is then determined by:

$$SSE = \sum [p_i - \hat{p}_i]^2$$

The same equations are applied to the CD algorithm to determine the least squares line and compute the SSE based on the CD data set (CD(p,t)).

4.4 Experimental Results

The same experimental setup as in chapter 3 was used to prove the validity of this cooperative windowing approach. The XY table was programmed to move the object in the same triangular path (Figure 3.3), except that the lamp used was moved in a way such that the projected light intensity was no longer the same in all of the areas of the path.

Before implementing the window confidence measure within the Foveal windows, an initial experiment was performed to establish the need for COOP. In this experiment, the NCC and CD results were compared to the actual current position of the object over the entire 300 point path, and the position of the Foveal windows were updated based on the result closest to the actual position. The NCC template was acquired with an average gray value of 71, and the selected CD threshold TH was 70. The average gray value of

the Foveal windows did not remain constant over the entire path as shown in Figure 4.2. The CD algorithm was applied in Fw1 (window 3), and the NCC algorithm was applied in Fw2 (window 4). The Foveal windows coordinates alternated between those calculated in Fw1 and Fw2 (Figure 4.3), depending on the error of the computed target position. Since the interval between path points 50 and 250 exhibited an increase in the average gray value of the Foveal windows (Figure 4.2), the Foveal window coordinate switching process was expected to be dominated by window 3 (Figure 4.3). However, because the gray value of the remaining intervals (0 to 50 and 250 to 300) was closer to the template gray value, window 4 dominated.

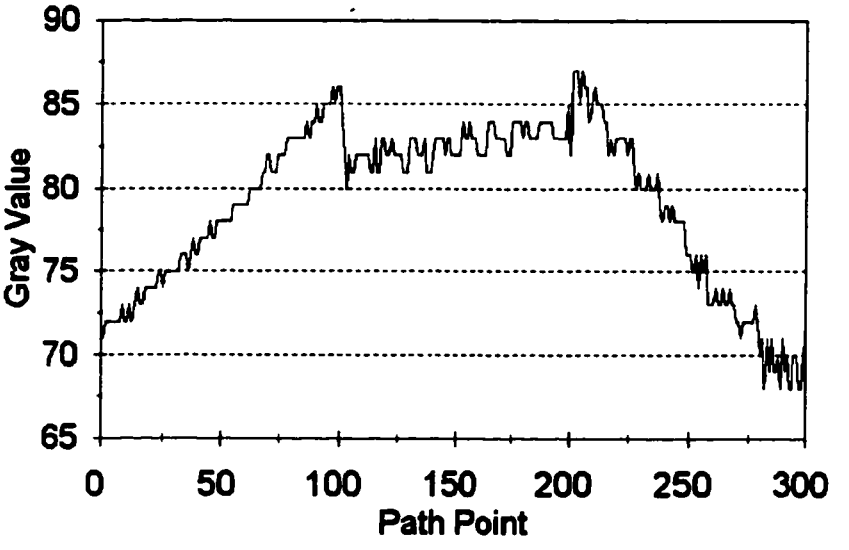


Figure 4.2. Average gray level intensity of the Foveal windows over the entire path.

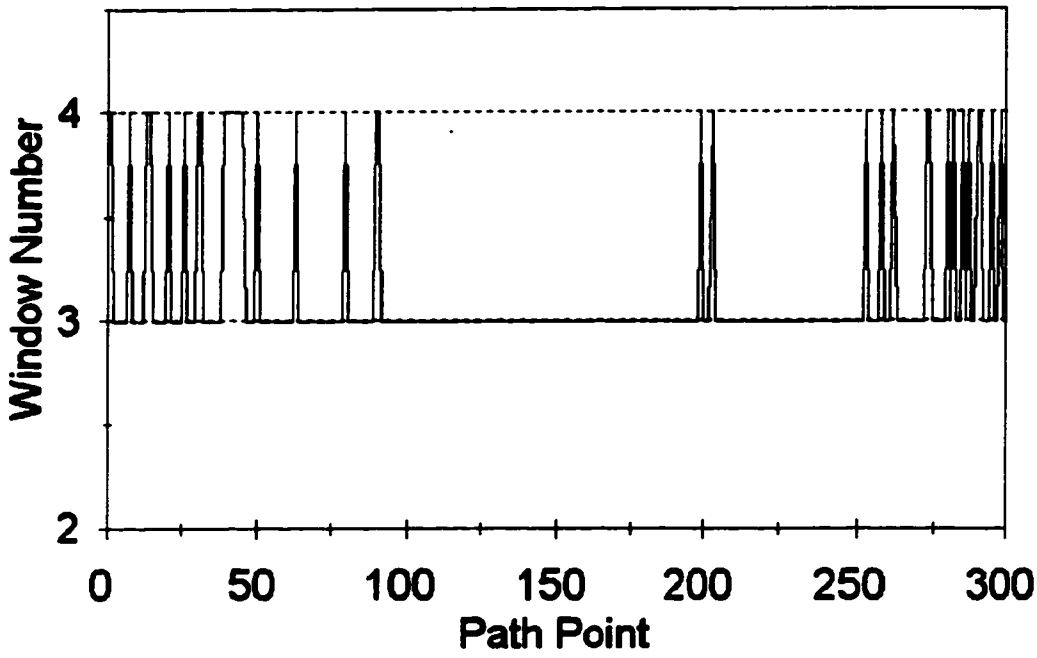


Figure 4.3. Switching process between the two Foveal windows.

The implementation of the proposed confidence measure in the Foveal windows verified the advantages of COOP. The SSE for the NCC and CD data sets were computed, and the Foveal windows coordinates were updated using the results of the algorithm with the lower SSE. The experiment was performed for data set sizes of three, four, and five samples. The average RMS position error were also computed for

comparison (Table 4.1), and the four sample COOP was selected because it resulted in the lowest error.

Method :	NCC	CD	COOP, data set size : 3 points, 4 points, 5 points
RMS error	0.63	0.97	0.61, 0.59, 0.60

Table 4.1. Average RMS position errors of the NCC, CD, and COOP methods.

The RMS position errors of the NCC, CD, and COOP methods are given in Figures 4.4, 4.5, and 4.6 respectively. The COOP algorithm is based on NCC in the intervals 0 to 100 and 200 to 300 (Figures 4.4 and 4.6). However, due to the higher error of NCC in the interval 100 to 200 (Figures 4.4 and 4.5), COOP is switched to use CD in this interval (Figures 4.5 and 4.6). Also, in this interval, the average gray level intensity of the Foveal windows (Figure 4.2) is almost 25% higher than the average gray level of the NCC template. Therefore, the NCC position error is expected to be higher, and using the CD algorithm is more reliable.

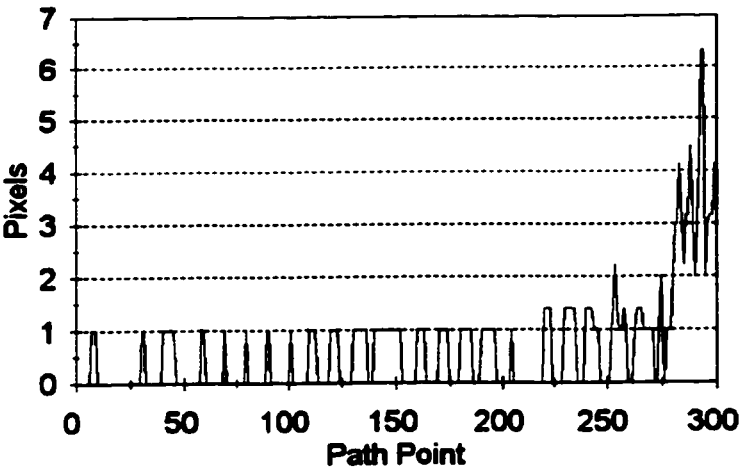


Figure 4.4. RMS position error using the NCC algorithm.

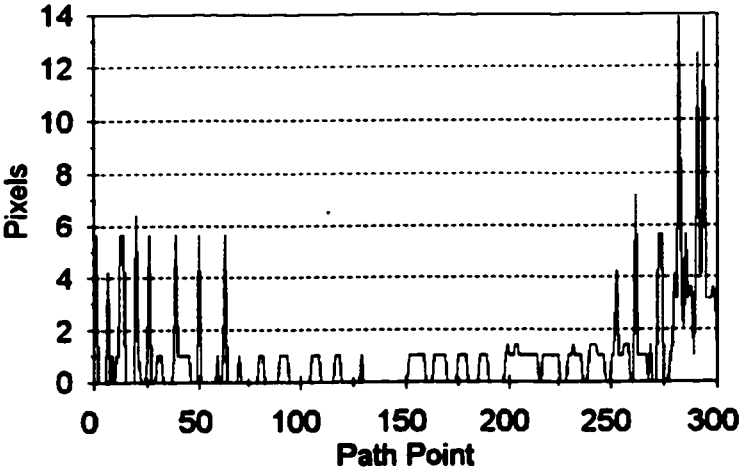


Figure 4.5. RMS position error using the CD algorithm.

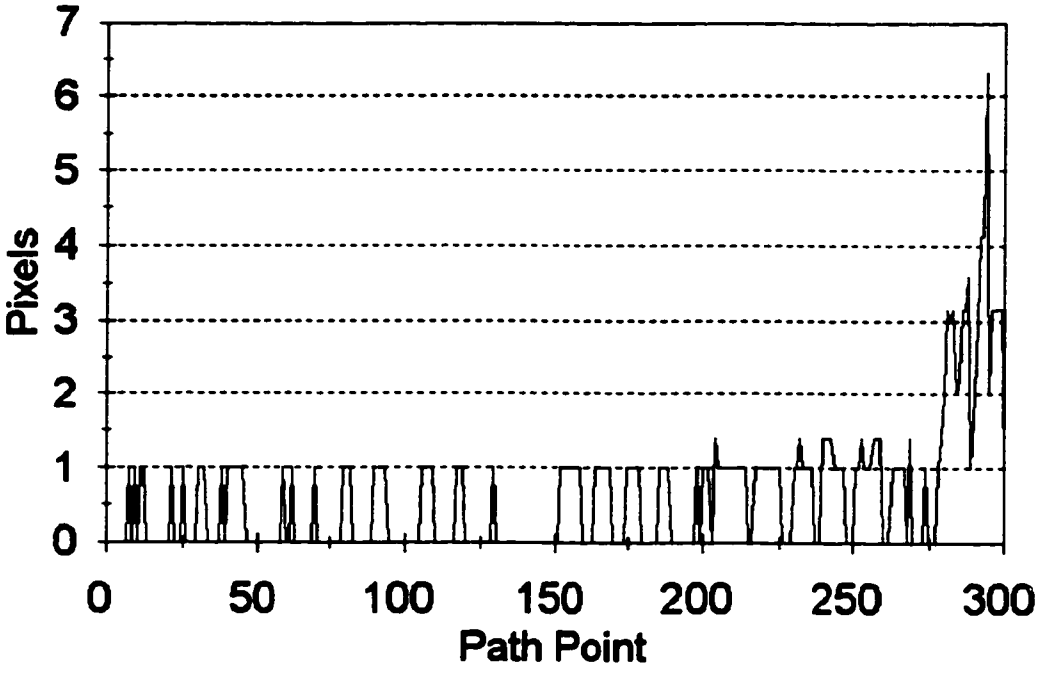


Figure 4.6. RMS position error using cooperative windowing.

CHAPTER 5

DISCUSSION

5.1 CONCLUSIONS

We have developed and implemented a flexible windowing network capable of acquiring multiple image windows of selectable size and position within the camera's FOV on a frame-by-frame basis. The operation of the windowing unit does not involve the host computer, thus avoiding delays associated with using the host computer's communication bus. This also eliminates the need to use a host computer of a certain type. The network is modular, so that the number of windowing units can be increased without major modifications.

Unlike other windowing networks which are custom-built, this network is based on off-the-shelf components, and it does not require any special image processing hardware. With the window logic implemented in programmable gate array technology (FPGA), improvements and design changes in the windowing logic are readily programmed.

This DSP-based network for real-time imaging applications has been tested successfully. The system is capable of grabbing regions of interest within the image frame at the hardware level. The sizes and locations of these windows can be updated on a frame-by-frame basis at frame rates, demonstrated as high as 114 frames per second. Window processors operate independently, but are connected by 20 Mb/s communication ports, so that multi-function windowing or cooperative windowing schemes can be implemented. A motion tracking experiment was set up to test the system and demonstrate consistent tracking of feature point(s) on a moving object. A motion tracking algorithm based on image differencing and template matching was implemented using a variety of windowing strategies to run the experiment. The results of the several experimental runs indicate that this vision system can be used successfully in applications requiring high speed motion tracking capabilities. Targets moving at speeds of up to 100 pixels per second may be tracked on a frame-by-frame basis, with the camera operating at a frame rate of 114 frames per second. Position updates can also be provided by the DSPs at the same rate.

A computationally efficient corner detection algorithm that does not involve line fitting or calculation of moments has been developed and implemented. Multiple experiments using real images of moving objects were performed to assess performance. It was found that the algorithm is capable of locating corners accurately ($\pm$ 0.25 mm) at a high frame rate (114 Hz). Furthermore, the performance is robust for a wide range of edge magnitude thresholds. A new information content measure has also been developed

and applied to evaluate the corner detection performance based on the number of edge pixels in the image window. The peak value of this measure corresponds to the number of edge pixels that yield the minimum corner location error.

A cooperative windowing scheme is developed based on a combination of template matching and corner detection methods. A confidence measure uses a least-squares regression strategy to determine the algorithm with lower variations in its results, so that it is selected for use by COOP. The experimental results have demonstrated the advantages of using COOP over NCC and CD.

5.2 Future Work

This thesis has demonstrated the cooperative windowing approach in target tracking applications, based on a change of illumination scenario. This method could also be applied in experiments involving measurable image clutter and occlusions. However, a more sophisticated window confidence measure, such as Kalman filtering could be used to decide whether the CD or NCC results should be selected. This is an avenue worth exploring to further demonstrate the advantages of using COOP.

This vision system is designed in such a way that it can be easily expanded to include additional windowing units to meet the requirements of the application. This multiple window scenario can be used to improve the tracking capabilities of the system,

or to reduce the computational cost of the vision algorithms used. Possible future applications include extracting 3D information from the acquired 2D images using inverse photogrammetry, the implementation of a foveal vision system with a head/eye platform, and automatic target recognition.

APPENDIX A
WINDOW LOGIC IMPLEMENTATION

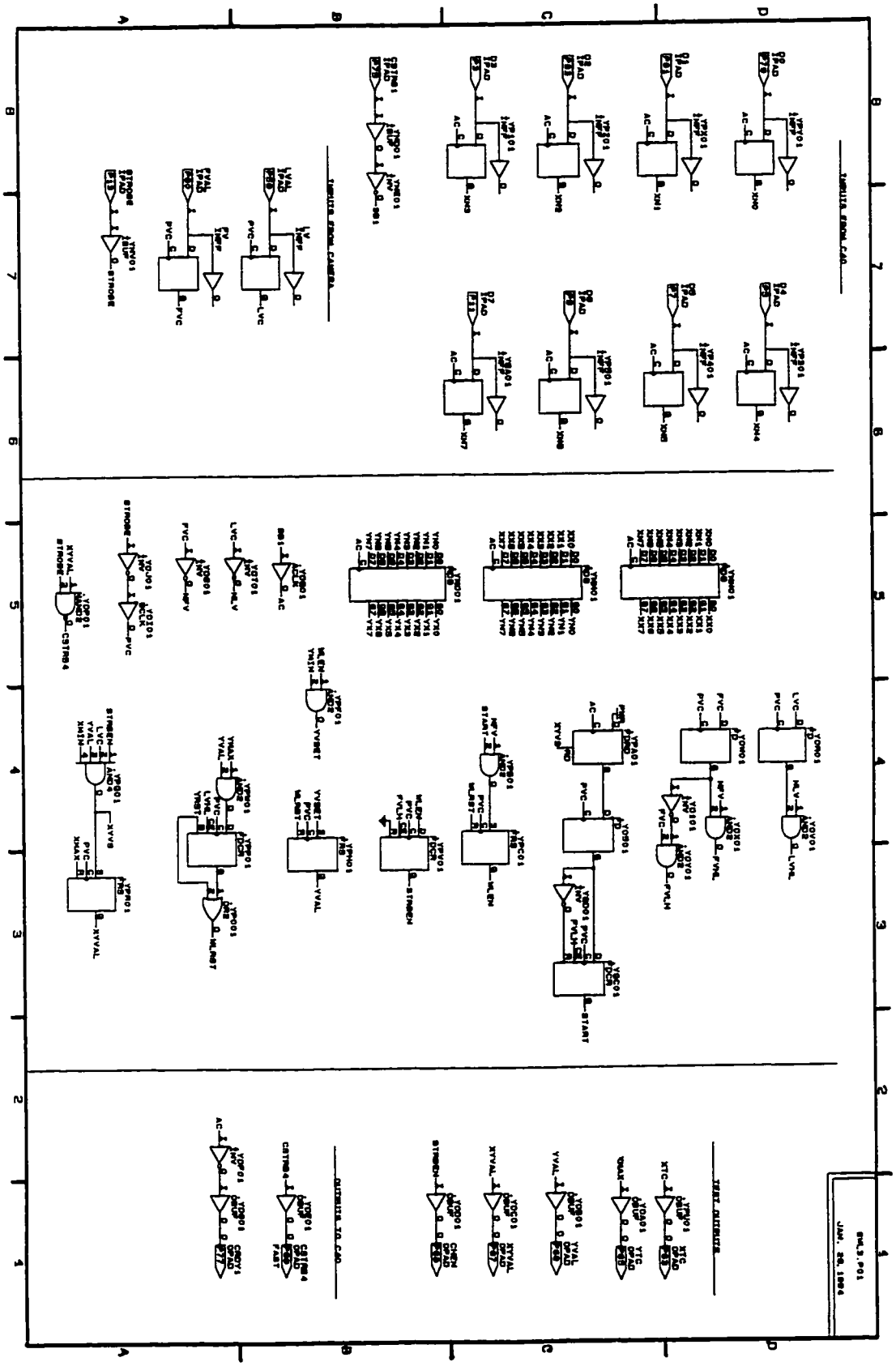


Figure A1.1. Window logic implementation (part 1).

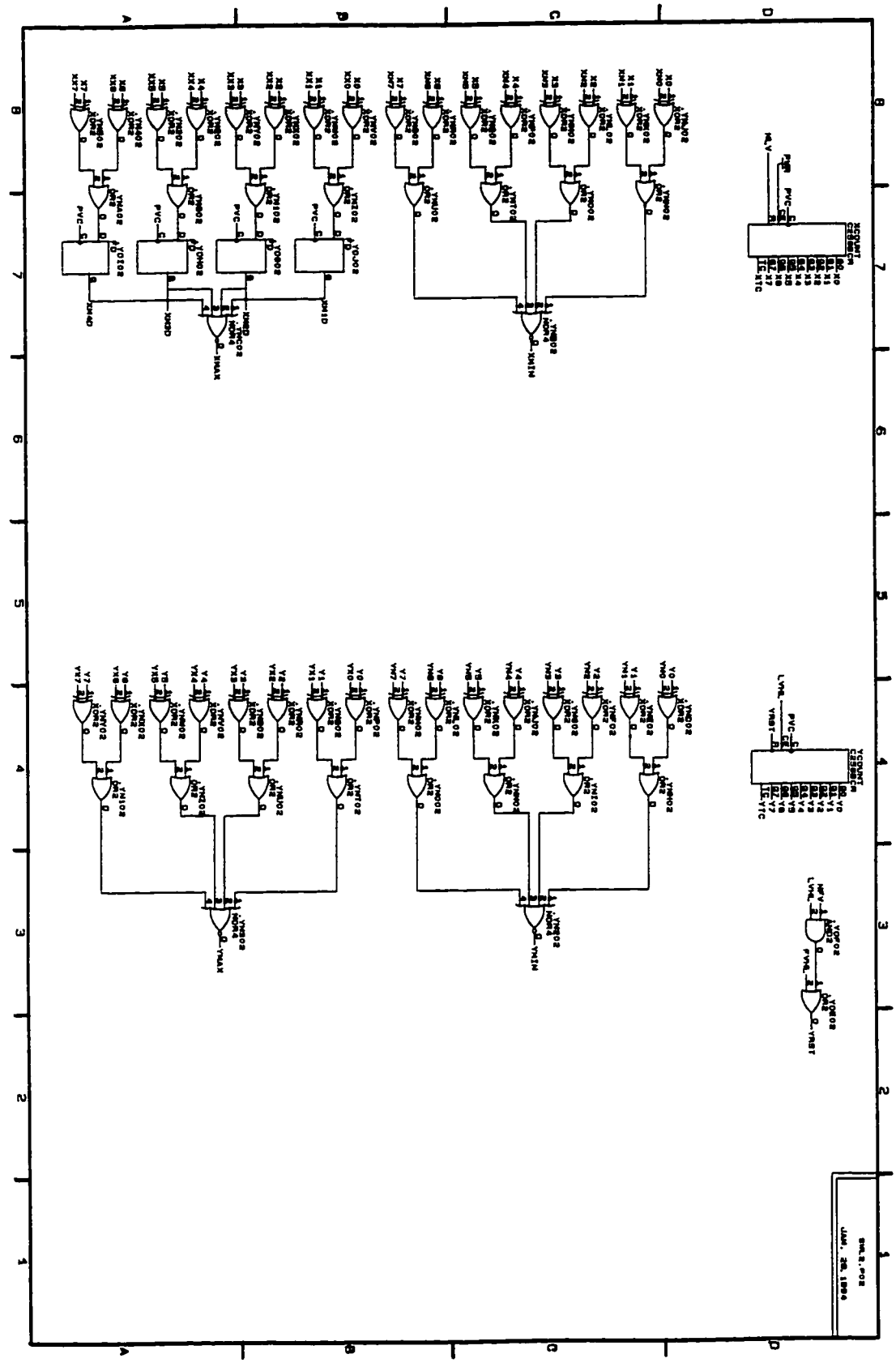


Figure A1.2. Window logic implementation (part 2).

AC	The SB1 signal is fed through the Auxiliary Clock buffer ACLK to latch the window coordinates in the shift registers, and set the Start flag.
FVC	Frame Valid synchronized to the PVC clock.
FVHL	Frame Valid High to Low transition flag.
FVLH	Frame Valid Low to High transition flag.
LVC	Line Valid synchronized to the PVC clock.
LVHL	Line Valid High to Low transition flag.
NFV	Not Frame Valid.
NLV	Not Line Valid.
PVC	The Not Pixel Valid signal is fed through the Global Clock buffer GCLK so that all internal operations are synchronized to this clock.
SB1	The C40's CSTRB1 signal is inverted to generate the SB1 clock.
START	A Start flag to begin executing the window logic operations. It is set once the window coordinates are strobed in.
STRBEN	A Strobe Enable flag which is set at the start of a new frame with WLEN already set.
WLEN	A Window Logic Enable flag which is set when NFV is high with START already set.
WLRST	A Window Logic Reset flag which is set either at the end of a frame, or at the end of all window logic operations.
X0-X7	The X count which represents the pixel number in a line of pixels.
XN0-XN7	An 8-bit number which represents the minimum X coordinate of the window.
XX0-XX7	An 8-bit number which represents the maximum X coordinate of the window.

Table A1.1. Xilinx internal signals definitions (Figure A1.1).

XMIN	The XMIN flag indicates that the X count is equal to the minimum X coordinate of the window.
XMAX	The XMAX flag indicates that the X count is equal to the maximum X coordinate of the window.
XTC	The X Terminal Count flag is set when the X count reaches 255.
XYVAL	The Window Valid flag XYVAL indicates that the current pixel is within the specified window coordinates.
Y0-Y7	The Y count which represents the number of lines in a frame.
YN0-YN7	An 8-bit number which represents the minimum Y coordinate of the window.
YX0-YX7	An 8-bit number which represents the maximum Y coordinate of the window.
YMAX	The YMAX flag indicates that the Y count is equal to the maximum Y coordinate of the window.
YMIN	The YMIN flag indicates that the Y count is equal to the minimum Y coordinate of the window.
YRST	The flag which resets the Y counter in the Not Frame Valid period.
YTC	The Y Terminal Count flag is set when the Y count reaches 255.
YVAL	The flag which indicates that the current line of pixels is within the specified window coordinates.

Table A1.2. Xilinx internal signals definitions (Figure A1.2).

APPENDIX B

WINDOW ACQUISITION TIMING DIAGRAMS

B1 Description of Timing Diagrams

B1.1 Start Flag Set (Figure B1.1)

1- The C40 strobes in the window coordinates into the Xilinx chip at an arbitrary point in time relative to the camera's synchronization signals. It is shown in this diagram to occur when the Frame Valid signal FVAL is low and the Not Frame Valid flag NFV is high, which represents the in-between frames period of invalid pixel data.

2- The window coordinates (xmin,xmax,ymin,ymax) are stored in (XN0-XN7), (XX0-XX7), (YN0-YN7), and (YX0-YX7) respectively. The input to the Start flag flip-flop (not shown) is synchronized to the PVC clock and then set.

3- A Line Valid high to low transition indicating the end of a line of invalid pixel data.

4- The Line Valid high to low transition is synchronized to the PVC clock. The Line Valid high to low transition flag LVHL is set, which enables the Start flag flip-flop.

5- The Start flag START is set, and LVHL is reset. The input of the Enable flip-flop is high if the NFV and START flags are both set.

6- The Enable flag WLEN is set.

7- The Start flag is reset one clock period after the Enable flag is set.

B1.2 Frame Valid Rising (Figure B1.2)

8- A Line Valid high to low transition indicating the end of a line, and a Frame Valid low to high transition indicating the beginning of a new frame. The Enable flag WLEN, which is an input to the Strobe Enable flip-flop, is already set.

9- The Line Valid and Frame Valid signals are both synchronized to the PVC clock. The Frame Valid low to high transition flag FVLH and the Line Valid high to low transition flag LVHL are set. The FVLH flag enables the Strobe Enable flip-flop, and the LVHL flag enables the Y counter YCOUNT.

10- The Strobe Enable flag STRBEN is set. The Y count becomes 1 and the YMIN flag is set, which indicates that the Y count is equal to the ymin window coordinate. The LVHL and FVLH flags are both reset. The LVC signal is low, so the Not Line Valid signal NLV is high, which resets the X counter XCOUNT.

11- The YVAL flag is set since the YMIN and WLEN flags are both high.

B1.3 Line Valid Rising (Figure B1.3)

12- A Line Valid low to high transition with FVAL high indicates the start of a new line of valid pixels. The STRBEN and YVAL flags are both high.

13- The Line Valid low to high transition is synchronized to the PVC clock. The X count still stands at zero, and the XMIN flag is high, which indicates that the X count is equal to the xmin window coordinate.

14- The Valid Pixel flag XYVAL is set. The X count is 1 and the XMIN flag is reset.

15- The pixel strobe signal CSTRB4 is activated and pulled low, and the first pixel is strobed into the memory of the C40.

16- The X count is 2, and the next pixel to be strobed in is pixel number 2.

The first pixel strobed in actually corresponds to the third valid pixel in the line. The digital data of the camera has to be delayed by two pixel clocks so that the first pixel strobed in does correspond to the first valid pixel in the line.

B1.4 Line Valid Falling (Figure B1.4)

17- Pixel number 253 has just been strobed in, and the X count is 254.

18- Pixel number 254 has just been strobed in, and the X count is 255, which sets the X counter's terminal count flag XTC. The X count is now equal to the xmax window coordinate, but the XMAX flag is set one clock period later.

19- A Line Valid high to low transition indicating the end of a line of valid pixels.

20- The Line Valid high to low transition is synchronized to the PVC clock, and the NLV flag is set high. Pixel number 255 has just been strobed in, and the X count is back to zero, which resets the XTC flag. The XMAX flag is set to indicate that the next pixel is the final valid pixel to be strobed in.

21- Pixel number 256 has just been strobed in, and the XYVAL flag is reset by the XMAX flag, which signals the end of valid pixels in this line of data. The pixel

strobe signal is deactivated and set high, and the XMAX flag is reset.

B1.5 Frame Valid Falling (Figure B1.5)

22- A Line Valid high to low transition indicating the end of the of a line, and a Frame Valid high to low transition indicating the end of a frame. The Y count is 256.

23- Both the Line Valid and Frame Valid signals are synchronized to the PVC clock, and the high to low transition flags FVHL and LVHL are set. The Y counter reset flag YRST is activated if FVHL is set or LVHL and NFV are both high. The Enable Reset flag WLRST is activated if YRST is high or YMAX,YVAL, and LVHL are all high.

24- The WLRST flag resets both the WLEN and YVAL flags, and the window logic is disabled.

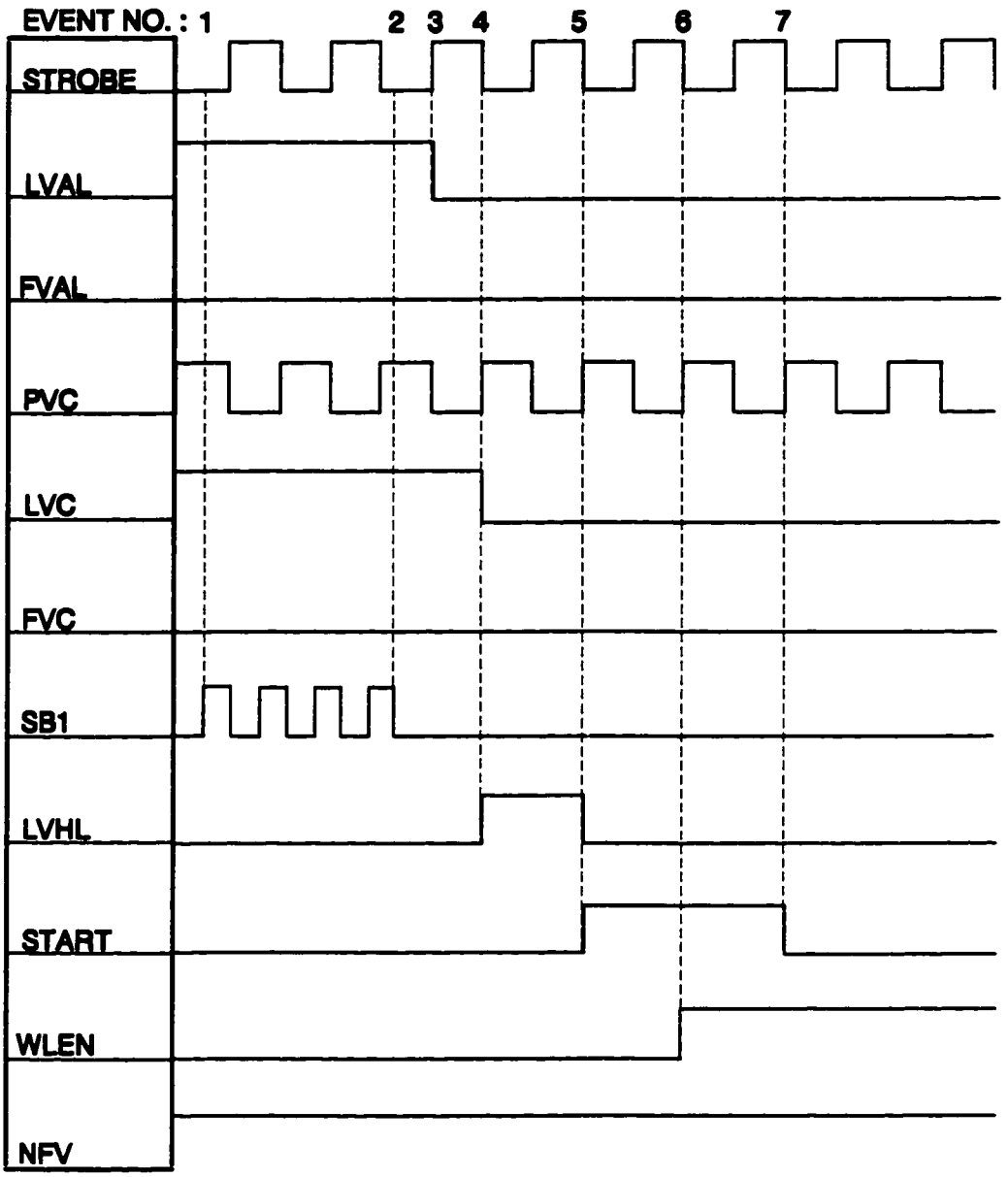


Figure B1.1. Timing diagram of the "Start Flag Set" sequence of events.

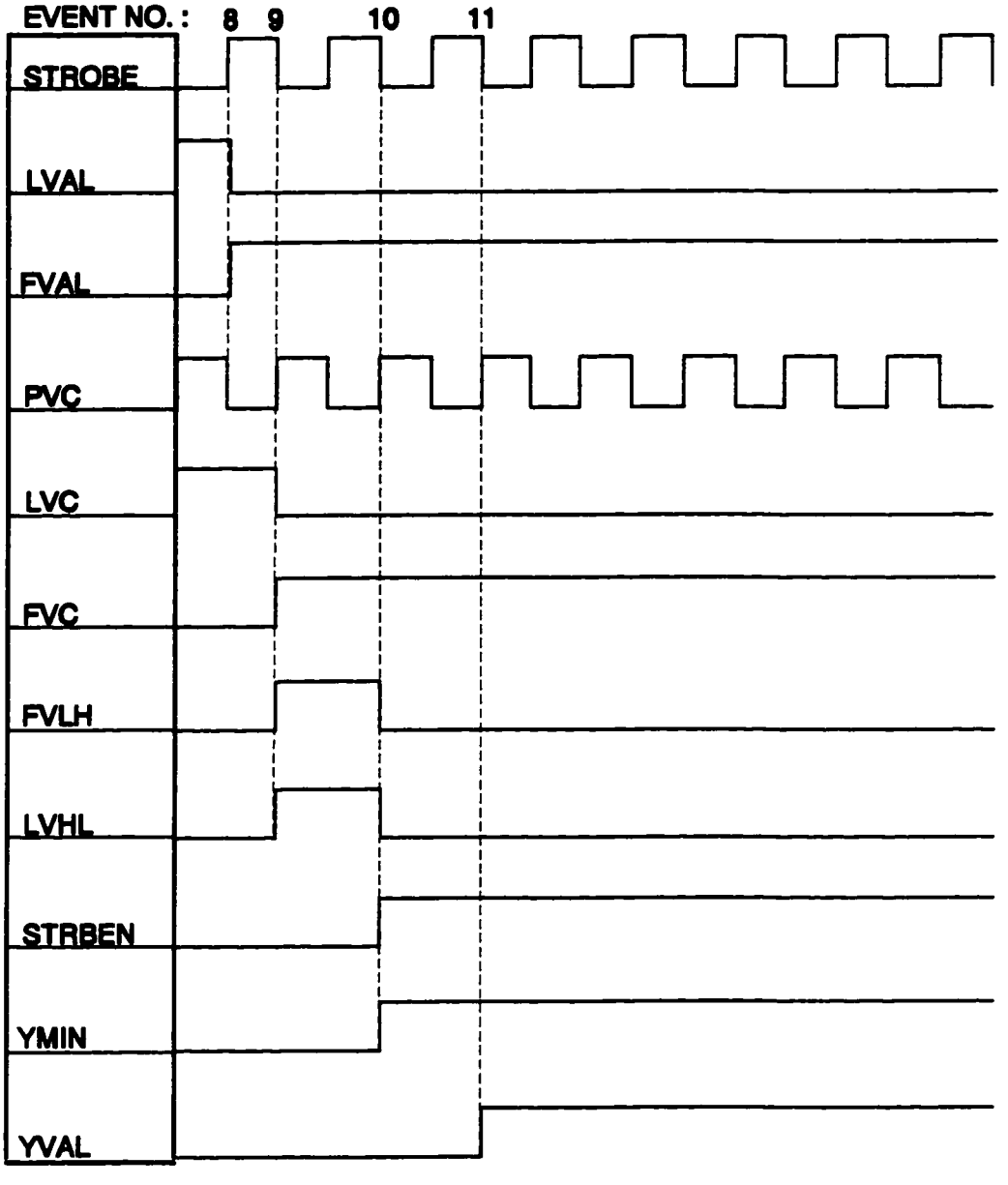


Figure B1.2. Timing diagram of the "Frame Valid Rising" sequence of events.

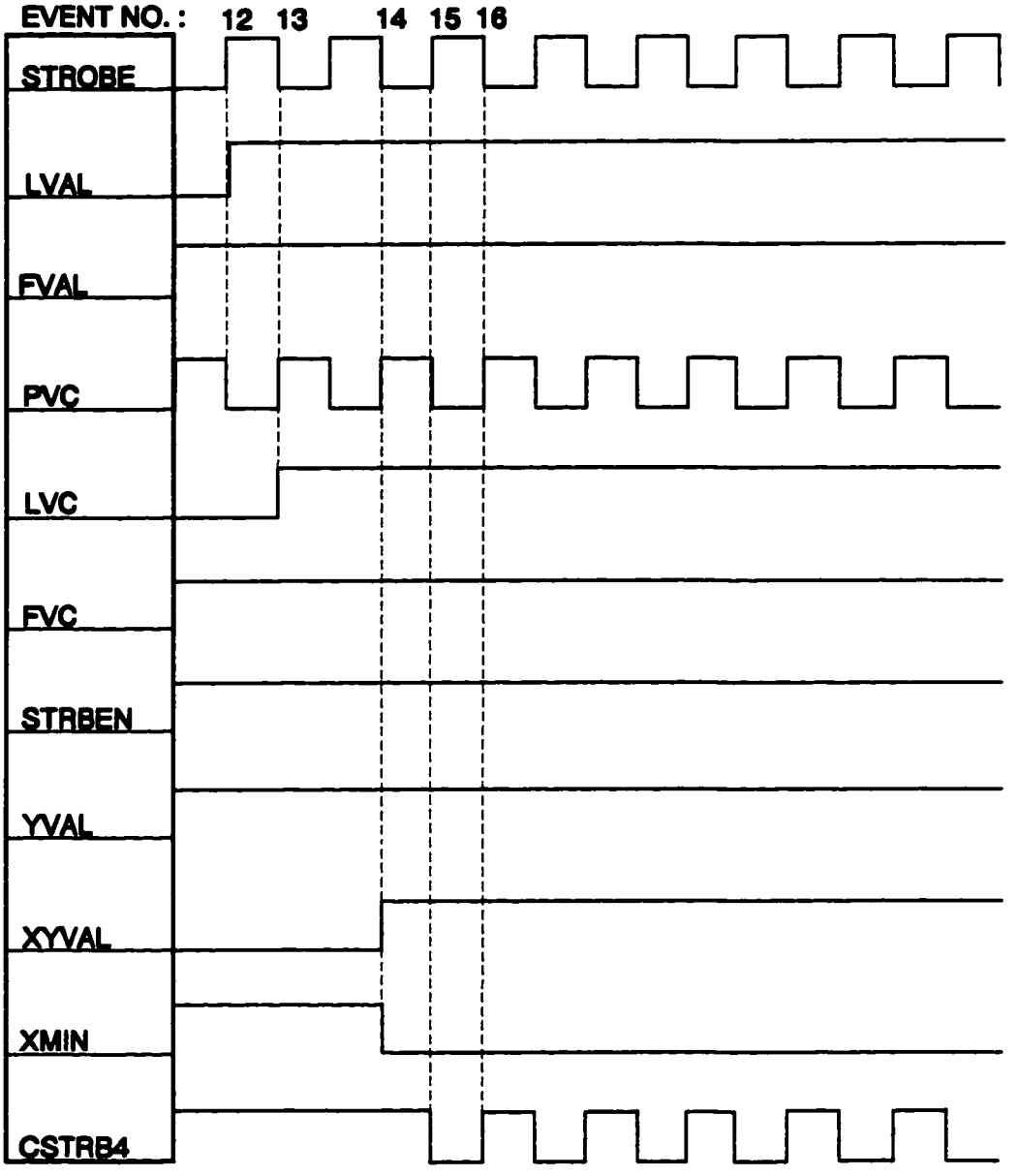


Figure B1.3. Timing diagram of the "Line Valid Rising" sequence of events.

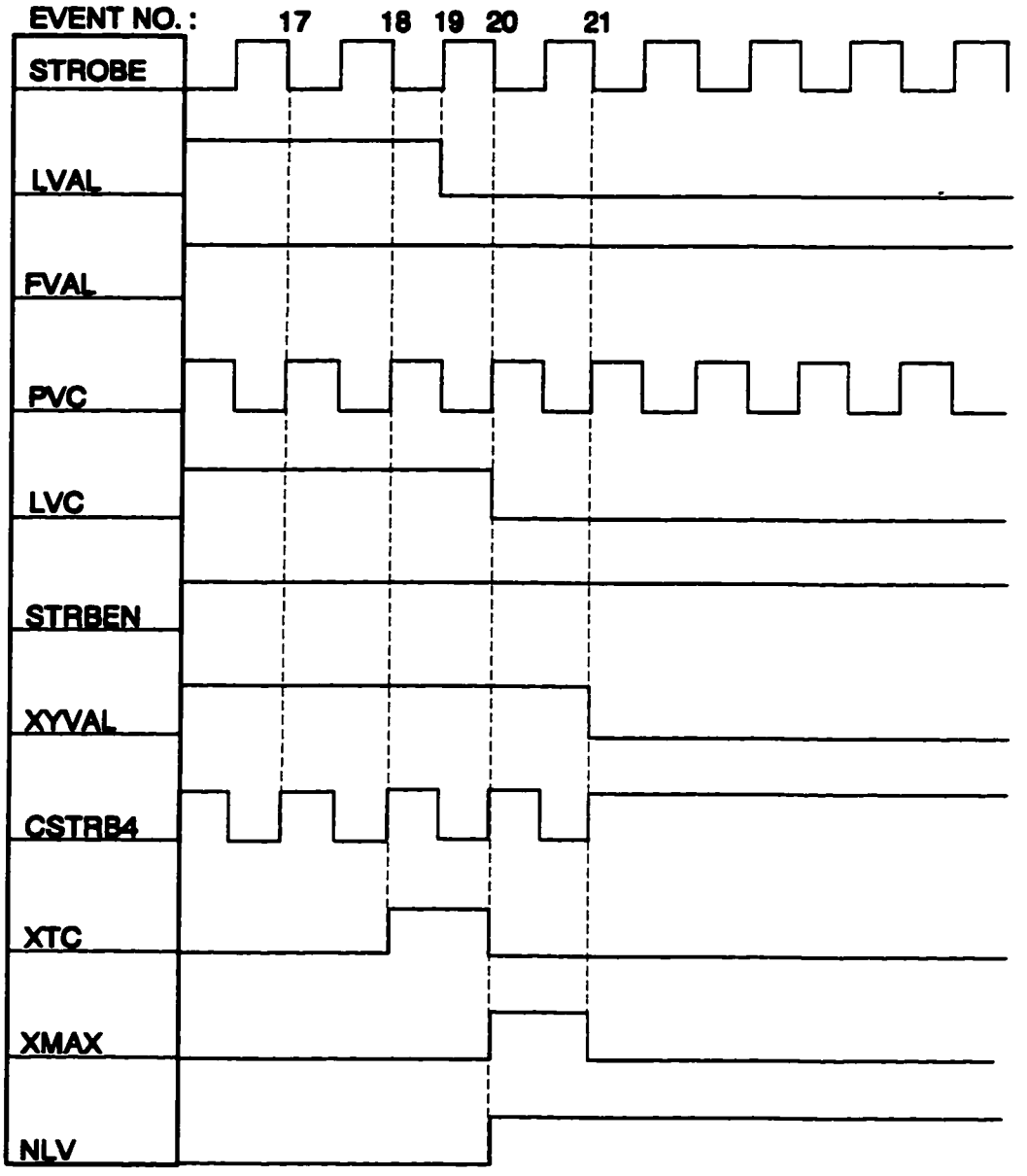


Figure B1.4. Timing diagram of the "Line Valid Falling" sequence of events.

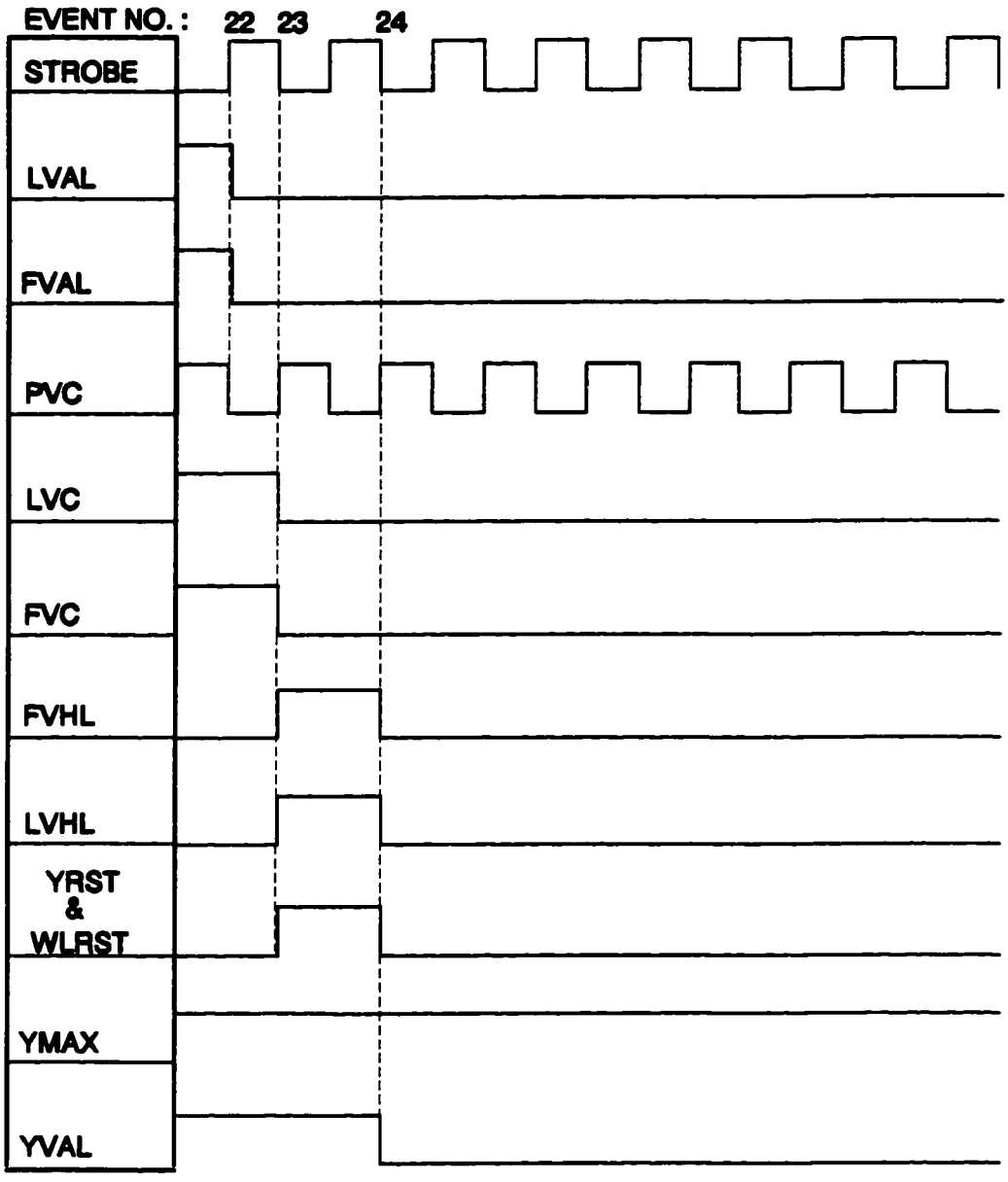


Figure B1.5. Timing diagram of the "Frame Valid Falling" sequence of events.

REFERENCES

[Aggarwal88] Aggarwal, J.K., Nandhakumar, N., "On the computation of motion from sequences of images -- a review", IEEE Proceedings, 1988, Vol. 76, No. 8, pp. 917-935.

[Allen93] Allen, P.K., Timcenko, A., Yoshimi, B., Michelman, P.,"Automated Tracking and Grasping of a Moving Object with a Robotic Hand-Eye System", IEEE Trans. on Robotics and Automation, Vol.9, No.2, p.152, April 1993.

[Anandan87] Anandan, P., "Measuring Visual Motion from Image Sequences", COINS Dept., Univ. of Massatchusetts, Tech. Rep. COINS-TR-87-21, 1987.

[Arch88] Archibald, C.C., "Real-Time Feedback Control Using a Laser Range Finder and Harmony", Proc. 7th Canadian CAD/CAM Robotics Conf., 1988, p. 6:56.

[Arking78] Arking, A., Lo, R.C., Rosenfeld, A., "A Fourier Approach to Cloud Motion Estimation", Journal of Applied Meteorology, 1978, Vol. 17, pp. 735-744.

[Aschwan92] Aschwanden, P., Guggenbuhl, W., "Experimental Results from a Comparative Study on Correlation-Type Registration Algorithms", Robust Computer Vision, Forstner/Rudwiedel (Eds), Wichmann 1992, pp. 268-289.

[Ayache91] Ayache, N., Lustman, F., "Trinocular Stereo Vision for Robotics", IEEE Trans. PAMI, Vol. 13, No.1, January 1991, pp. 73-85.

[Ayache87] Ayache, N., Lustman, F., "Fast and Reliable Passive Trinocular Stereovision", 1st Int. Conf. Computer Vision, June 1987, pp. 422-427.

[Barnard80] Barnard, S.T., Thompson, W.B., "Disparity Analysis of Images", IEEE Trans. PAMI, Vol. 2, No. 4, 1980, pp. 333-340

[Baron94] Baron, T., Levine, M.D., Hayward, V., Bolduc, M., Grant, D., "A Biologically-Motivated Robot Eye System", 8th CASI Conference on Astronautics, November 1994, pp. 231-240.

[Bennam94] Bennamoun, M., Boashash, B., "A Vision System for Automatic Object Recognition", 1994 IEEE Int. Conf. on Systems, Man, and Cybernetics, pp. 1369-1374.

[Bhanu86] Bhanu, B., "Automatic Target Recognition: State of the Art Survey", IEEE Trans. on Aerospace and Electronic Systems, Vol. 22, No. 4, July 1986, pp. 364-379.

[Buttazzo94] Buttazzo, G.C., et al., "Mousebuster: A Robot for Real-Time Catching", IEEE Control Systems, February 1994, pp. 49-56.

[Castano94] Castano, A., Hutchinson, S., "Visual Compliance: Task-Directed Visual Servo Control", IEEE Trans. on Robotics and Automation, Vol. 10, No. 3, June 1994, pp. 334-342.

[Corke93] Corke, P., "Visual Control of Robot manipulators -- A Review", K. Hashimoto ed., Vol. 7 of Robotics and Automated Systems, pp. 1-31, World Scientific.

[Dhond91] Dhond, U.R., Aggarwal, J.K., "A Cost-Benefit Analysis of a Third Camera for Stereo Correspondence", Int. J. Computer Vision., Vol. 6, No. 1, 1991, pp. 39-58.

[Dhond89] Dhond, U.R., Aggarwal, J.K., "Structure from Stereo: A Review", IEEE Trans. Syst. Man Cyber., Vol. 19, No. 6, 1989, pp. 1489-1510.

[Feddema89] Feddema, J., Mitchell, O., "Vision-Guided Servoing with Feature Based Trajectory Generation", IEEE Trans. on Robotics and Automation, Vol.5, No.6, p.691, 1989.

[Frendo89] Frendo, M.J., "Three Dimensional Tracking of Four Point Planar Patterns Using Corners", Ph.D. Thesis, McMaster University, 1989.

[Fukui92] Fukui, K., Nakai, H., Kuno, Y., "Multiple Object Tracking System with Three Level Continuous Processes", IEEE 3rd Int. Conf. on Computer Vision, 1992, pp. 19-27.

[Gaiarsa94] Gaiarsa, A.E., Capson, D.W., "Real-Time Measurement of Corner Position in Binary Images", IEEE Trans. on Instrumentation and

Measurement, Vol. 43, No. 4, August 1994, pp. 567-577.

[Gennery87] Gennery, D., "Sensing and Perception Research for Space Telerobotics at JPL", Proc. of the 1987 IEEE Conference on Robotics and Automation, p.311, 1987.

[Graefe84] Graefe, V., "Two Multi-Processor Systems for Real-Time Vision", Robotics and Artificial Intelligence, M. Brady et al. (eds), Springer-Verlag Berlin Heidelberg 1984, pp. 301-308.

[Hager96] Hager, G.D., Hutchinson, S., Corke, P., "Visual Servo Control", Tutorial TT3, IEEE Int. Conf. on Robotics and Automation, April 1996.

[Haralick93] Haralick, R.M., Shapiro, L.G., Computer and Robot Vision, Vol. I, 1993, Addison-Wesley, pp. 410-419.

[Hill79] Hill, J., Park, W.T., "Real-Time Control of a Robot with a Mobile Camera",Proc. 9th ISIR, March 1979, pp. 233-246.

[Horn81] Horn, B.K.P., Schunk B.G., "Determining Optical Flow", Artificial Intelligence, 1981, pp. 185-203.

[Horn87] Horn, B.K.P., "Motion Fields are Hardly Ever Ambiguous", Int. J. Computer Vision, Vol. 1, pp. 263-278, 1987

[Huang94] Huang, T.S., Netravali, A.N., "Motion and Structure from Feature Correspondences: A Review", Proceedings of the IEEE, Vol. 82, No. 2, 1994.

[Huertas81] Huertas, A., "Corner Detection for Finding Buildings in Aerial Images", USCIPI Report 1050, University of Southern California, 1981, pp. 61-68.

[Hussain91] Hussain, Z., Digital Image Processing/Practical Applications of Parallel Processing Techniques, Ellis Horwood 1991, pp. 141-143.

[Hutch96] Hutchinson, S., Hager, G.D., Corke, P.I., "A Tutorial on Visual Servo Control", IEEE Trans. on Robotics and Automation, Vol.12, No.5, October 1996, pp. 651-670.

[Inoue85] Inoue, H., Mizoguchi, H., "A Flexible Multi Window System for Robots", 2nd International Symposium on Robotics Research (1985), Cambridge, Mass., pp. 95-102.

[Ito86] Ito, M., Ishii, A., "Range and Shape Measurement Using Three-View Stereo Analysis", IEEE Conf. Comp. Vis. Patt. Recog., June 1986, pp. 9-14.

[Jain95a] Jain, R., Kasturi, R., Schunck, B.G., Machine Vision, 1995, McGraw-Hill, pp. 194-214.

[Jain95b] Jain, R., Kasturi, R., Schunck, B.G., Machine Vision, 1995, McGraw-Hill, pp. 61-69.

[Jain81] Jain, R., "Dynamic Scene Analysis Using Pixel-Based Processes", IEEE Computer, August 1981, pp. 12-18.

[Koivo91a] Koivo, A.J., Houshangi, N., "Real-Time Vision Feedback for Servoing Robotic Manipulator with Self-Tuning Controller", IEEE Trans. on Systems,Man,and Cybernetics, Vol.12, No.1, January 1991, pp. 134-141.

[Koivo91b] Koivo, A.J., "On Adaptive Vision Feedback Control of Robotic Manipulators", IEEE Conf. Des. Cont., December 1991, pp. 1883-1888.

[Kubota93] Kubota, H., Okamoto, Y., Mizogushi, H., Kuno, Y., "Vision Processor System for Moving-Object Analysis", Machine Vision and Applications 1993, 7:37-43.

[Lang87] Lang, G.K., Gale, M.T., Knop, K., "A Low Cost Smart Camera and its Application to motion Detection and Surveillance", Time-Varying Image Processing and Moving Object Recognition, V. Cappellini (ed.), Elsevier Science Publishers B.V., 1987.

[Lee92] Lee, K.M., Blenis, R., "Flexible, Integrated Machine Vision", Vision, Vol. 8, No. 4, 1992.

[Matties89] Matties, L., Kanade, T., Szeliski, R., "Kalman Filter-Based Algorithms for Estimating Depth from Image Sequences", Int. J. Computer Vision, Vol. 3, pp. 209-236, 1989.

[Murray94] Murray, D., Basu, A., "Motion Tracking with an Active Camera", IEEE Trans. on PAMI, Vol.16, No.5, May 1994, pp. 449-459.

[Murray93] Murray, D.W., McLauchlan, P.F., Reid, I.D., Sharkey, P.M., "Reactions to Peripheral Image Motion using a Head/Eye Platform", IEEE 4th Int. Conf. on Computer Vision, December 1993, pp. 403-411.

[Nagel87] Nagel, H.H., "On the Estimation of Optical Flow: Relations between Different Approaches and Some New Results", Artificial Intelligence, Vol. 33, 1987, pp. 299-324.

[Pap91] Papanikolopoulos, N.P., Khosla, P.K., "Feature Based Robotic Visual Tracking of 3-D Translational Motion", IEEE Conf. Des. Cont., December 1991, pp. 1877-1882.

[Pap93] Papanikolopoulos, N.P., Khosla, P.K., Kanade, T., "Visual Tracking of a moving Target by a Camera Mounted on a Robot: A Combination of Control and Vision", IEEE Trans. on Robotics and Automation, Vol.9, No.1, p.14, February 1993.

[Peters88] Peters, R.A.II, "Image Complexity Measurement for Predicting Target Detectability", Ph.D. Thesis, University of Arizona, 1988.

[Pietikan86] Pietikanien, M., Harwood, D., "Depth from Three Camera Stereo", IEEE Conf. Comp. Vis. Patt. Recog., June 1986, pp. 2-8.

[Press90] Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., Numerical Recipes in C, Cambridge, 1990, pp. 290-298.

[Reid93] Reid, I.D., Murray, D.W., "Tracking Foveated Corner Clusters Using Affine Structure", IEEE 4th Int. Conf. on Computer Vision, December 1993, pp. 76-83.

[Rizzi92] Rizzi, A.A., Whitcomb, L.L., and Kodischek, D.E., "Distributed Real-Time Control of a Spatial Robot Juggler", IEEE Computer, May 1992, pp. 12-24.

[Sadjadi92] Sadjadi, F., "Automatic Recognition of Partially Occluded Objects", Proceedings of the SPIE, Automatic Target Recognition II, Vol. 1700, 1992, pp. 277-284.

[Schunck84] Schunck, B.G., "The Motion Constraint Equation for Optical Flow", Proc. 7th Int. Conf. on Pattern Recognition, 1984, Montreal, Canada, pp. 20-22.

[Sethi87] Sethi, I.K., Jain, R., "Finding trajectories of Feature Points in a Monocular Image Sequence", IEEE Trans. on PAMI, Vol. 9, No. 1, January 1987, pp. 56-73.

[Shah84] Shah, M.A., Jain, R., "Detecting Time-Varying Corners", Proc. 7th Int. Conf. on Pattern Recognition, 1984, Montreal, Canada, pp. 2-5.

[Silven93] Silven, O., Repo, T., "Experiments with Monocular Visual Tracking and Environment Modeling", 4th Int. Conf. on Computer Vision, 1993, pp. 84-92.

[Sobel70] Sobel, I., "Camera Models and Machine Perception", Stanford AI Memo 121, May 1970.

[Thom79] Thompson, W.B., "Combining Motion and Contrast for Segmentation", Comp. Sci. Dept., Univ. of Minnesota, March 1979.

[Tian84] Tian, Q., Huhns, M.N., "A Fast Iterative Hill Climbing Algorithm for Subpixel Registration", Proc. 7th Int. Conf. Patt. Recog., Montreal, Canada, 1984, pp. 13-15.

[Vega89] Vega-Riveros, J.F., Jabbour, K., IEE Proceedings, Vol. 136, Pt. I, No. 6, December 1989, pp. 397-404.

[Venkat90] Venkatesan, S., Archibald, C., "Real-Time Tracking in Five Degrees of Freedom Using Two Wrist-mounted Laser Range Finders", IEEE Intl. Conf. on Robotics and Automation, Cincinnati, Ohio, May 1990, pp. 2004-2010.

[Wang91] Wang, J., Wilson, W.J., "3D Relative Position and Orientation Estimation Using Kalman Filtering for Robot Control", IEEE Robotics and Automation Conf., Nice, France, May 10-15, 1992, pp. 2638-2645.

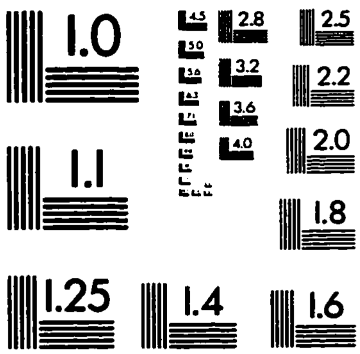
[Wang94] Wang, H., Brady, M., "A Practical Solution to Corner Detection", IEEE Int. Conf. on Image Processing, 1994, Vol. I, pp. 919-923.

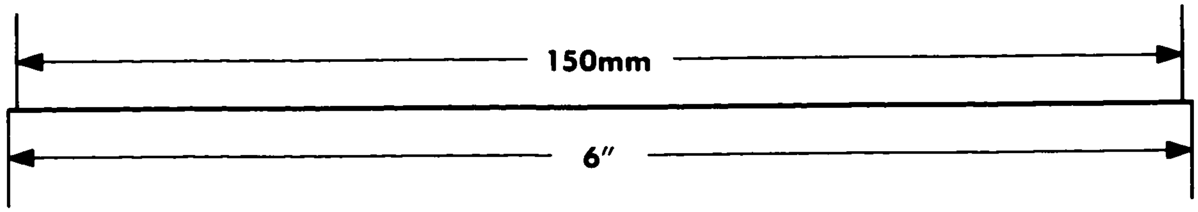
[Weiss87] Weiss, L.E., Sanderson, A.C., Neuman, C.P., "Dynamic Sensor-Based Control of Robots with Visual Feedback", IEEE J. Robotics and Automation, Vol.3, No.5, October 1987, pp. 404-416.

[Wilson88] Wilson, W.J., editor. "Vision Sensor Integration for Dynamic Control of Robots", Robots 12 Vision'88 Conf., Detroit, Michigan, June 5-9, 1988.

[Yashida85] Yashida, M., "3-D Aquisition by Multiple Views", Proc. Int. Symp. Robotics Res., October 1985.

IMAGE EVALUATION TEST TARGET (QA-3)





APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989