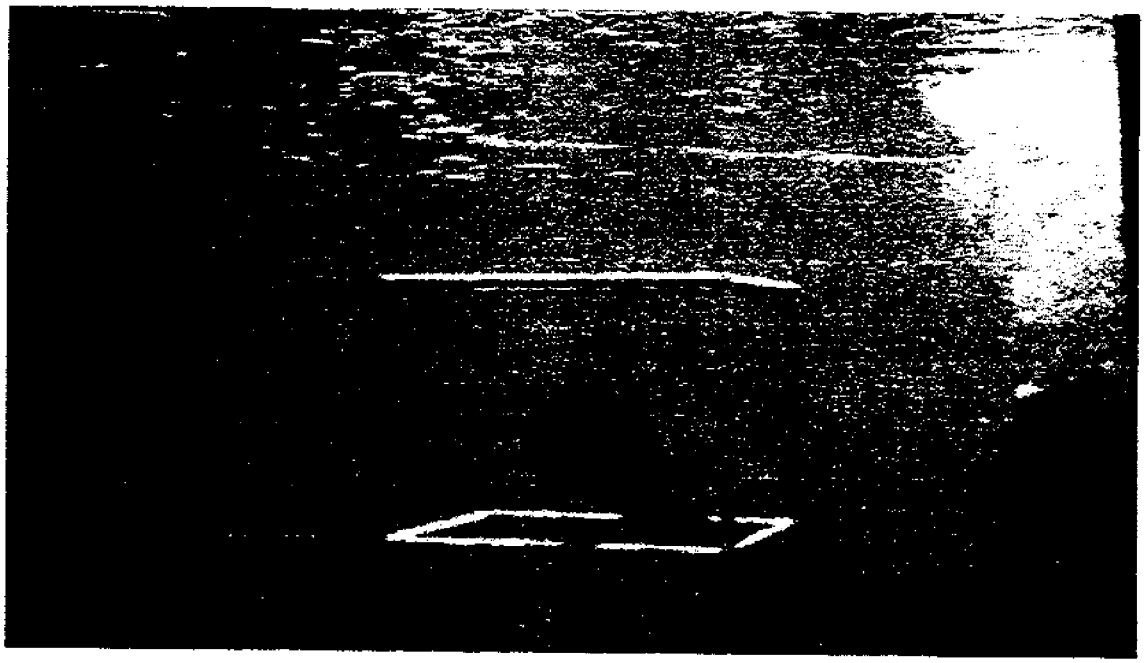
Optical Positioning Instrumentation & Evaluation



University of New Hampshire
Ocean Research Projects
TECH 797

Project Advisors:
Dr. Kenneth C. Baldwin
Dr. M. Robinson Swift

OPIE Team:
Derek Michelin
Shannon Stott

Sea Grant
New Hampshire/Maine

Abstract

Analyzing the response of marine structures to controlled wave simulations provides a great aid in improving their design and defining their performance. The need for accurate, planar motion analysis of models in the University of New Hampshire Center for Ocean Engineering wave tank prompted the design of OPIE, an *o*ptical *p*ositioning *i*nstrumentation and *e*valuation system. Extensive research was conducted to identify the optimal system type for this application. Optical analysis methods were the clear solution. The final OPIE system consisted of a CCD camera, frame grabber, and a custom designed computer. A powerful software package was developed to analyze the images and generate practical depiction of the models kinematics. OPIE has proven through initial tests to provide an accurate representation of the position, velocity and acceleration of planar motion. This system incorporates state-of-the-art technology with a user-friendly platform allowing for the easy implementation of advanced applications.

Acknowledgments

This work is the result of research sponsored in part, by the National Sea Grant College Program, NOAA, Department of Commerce, under grant #NA36RG0110 through the University of New Hampshire/University of Maine Sea Grant College Program.

We would like to extend our thanks to the following persons who contributed greatly to the success of OPIE:

Dr. Kenneth C. Baldwin
Dr. M. Robinson Swift
Dr. Larry Harris
Dr. David Watt
Joel Wright at InfoServe
Jim Rowell at JKN Electronics
Jon Scott
Linda MacPherson
Muriel Bunker
and the people of the Center for Ocean Engineering.

Table of Contents

List of Figures

List of Tables

I. Introduction

Marine structures such as ship hulls, buoys, and fish cages are all largely affected by motion of the ocean around them. A vessel's righting moment (resistivity to capsizing) and pitching moment (a major cause of seasickness and cargo shifting) are major concerns for the designer. Analyzing the response of physical models to a controlled wave simulation provides a great aid in improving a device's design and performance. The University of New Hampshire, Center for Ocean Engineering (COE) wave tank needed a proper system for evaluating the kinematics of the object under test (OUT). The goal of this project was to design a method of motion analysis for the wave tank from which position, velocity and acceleration information could be obtained for the OUT.

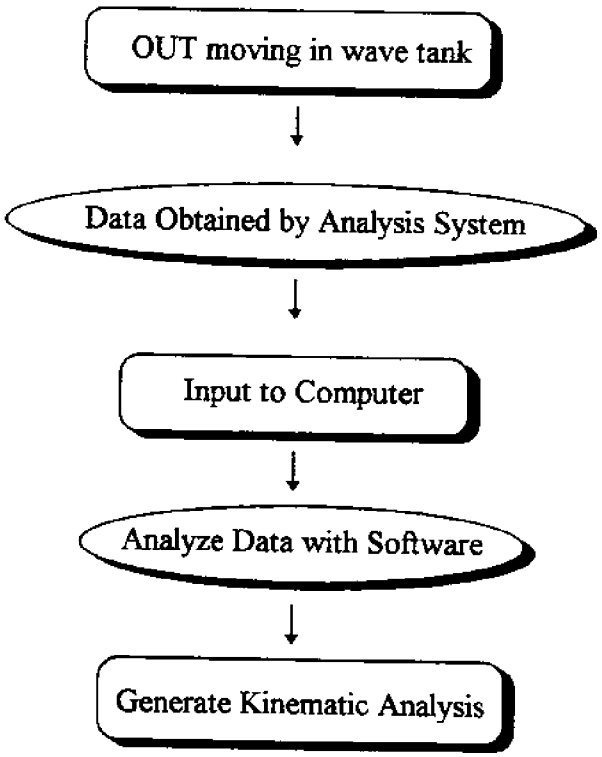
The problem was approached in three stages. The first stage consisted of extensive research into current methods used for motion analysis. Once a clear understanding of these systems was gained, a refined criteria for the OPIE (optical positioning instrumentation and evaluation) system was developed. During the second stage, specific components were carefully selected and the purchasing process began. Finally, the system implementation began, and a unique computer software program was written to provide the operations necessary to develop kinematic analyses.

II. Problem Specifications

Presented with the initial goals of the project, a flow chart (Figure 1) was developed to clearly define each step required for data acquisition and analysis.

The ideal system must be capable of accurately measuring the heave, pitch, roll and yaw of the model (see Figure 2) without disrupting the model's natural motion. The setup should use the latest technology, making it competitive with other tank motion analysis systems in the area. The optical positioning instrumentation and evaluation system (OPIE) was developed with the knowledge that the primary users of the system would be student researchers, professors and external corporations.

Figure 1: Flow Chart for Motion Analysis System



The following parameters were defined as requirements for its success:

Measurement Requirements: The system must provide measurements of two-dimensional planar motion with the possibility of upgrading to 3-dimensional analysis.

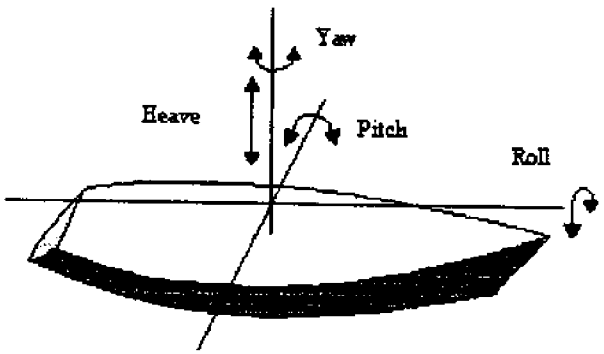


Figure 2: Motions a vessel may experience.

Heave: Motion parallel to the vertical axis.

Pitch: Rotation about a horizontal axis set perpendicular to the longitudinal axis.

Roll: Rotation about the longitudinal axis.

Yaw: Rotation about the vertical axis

Accurate: The system should provide data that allows for precise modeling of the OUT's kinematics.

Universal: The system should analyze any planar movement and have the ability to be modified for experiments outside of the wave tank.

Cost Efficient: The components should be of high quality, yet still remain within the project budget.

Competitive: State-of-the-art equipment which exceeds the present standards in performance available at other testing facilities. The components should allow flexibility for future upgrades.

Quick: The system output should be processed quickly and generate reliable results.

II. Research

At the onset of this project, the members of OPIE had minimal knowledge of the available techniques for motion analysis. The research began on the World Wide Web where after countless hours of searching documents, the system selection was narrowed down to three methods: electromagnetic, physical sensors and optical (i.e. cameras) systems.

A. Review of Data Acquisition Systems

Electro-Magnetic

Electro-magnetic analysis provide for wireless, accurate 3-Dimensional position analysis by measuring changes in magnetic flux. Electro-magnetic systems are typically used when no alternate systems exist because of their extremely high cost. Electro-magnetic systems can be found in kinesiology labs to measure human movement

Physical Sensors

Typical sensors are linear variable differential transformers (LVDT) and potentiometers. The LVDT produces an AC output which is proportional to a change in position. A potentiometer or an electrical resistance transducer also generates an output signal that is proportional to the change in position. While having good sensitivity and dynamic response, they inhibit the motion of the object by adding restraining forces and/or mass. Physical sensors are used in quasi-static applications in machine shops.

Optical

An optical system typically consists of a camera and computer assembly which captures add processes video images of the object's motion. The camera allows for high data acquisition rates and ample resolution. The camera is non-intrusive, providing an accurate representation of the object's movement. However, the speed at which data can be acquired from a camera is limited by the computer. Software selection can also limit the scope of analysis. Optical systems are commonly used to analyze motion; its present applications range from analyzing golf swings to vehicle crash tests.

After comparing each system to the stated criteria, it was concluded that altering the response of the model to the waves would be a serious detriment to the accuracy of the analysis. The use of bulky objects, essential for LVDT and potentiometer systems, alters the moments of inertia and overall weight of the object. While the physical sensors are relatively inexpensive, the loss of a precise simulation was too valuable a commodity to sacrifice.

The sensors required by the electromagnetic system are very small and would not cause any interference with the natural motion of the object. However, the major pitfall of the electromagnetic system is its cost. The starting price for an electromagnetic system is around $50,000.

The setup which best satisfied all of the stated objectives was the optical system. This system is non-intrusive, accurate and had a manageable price. The type of optical system of interest consists of a camera, frame grabber and computer running suitable software.

B. Investigation of Optical Systems

Knowing that an optical system would be used, initial brainstorming efforts defined the specifics of the setup. The basic framework of a test would consist of an OUT moving in the wave tank with its motion recorded by a camera. The camera would be outside the wave tank and connected to a computer which would store the video sequences. (See Figure 3) The OUT would have two painted markers which contrasted with the model color and nearby

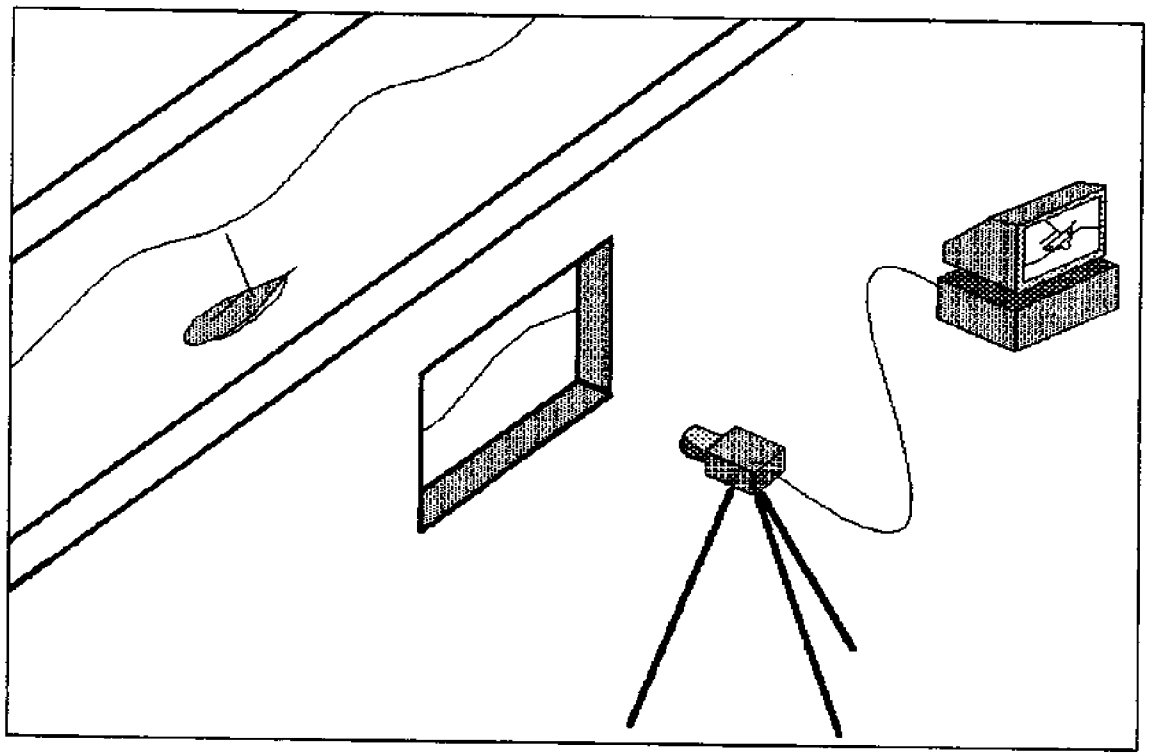


Figure 3: Optical System Configuration

background. After the video was recorded, the computer would analyze the video and obtain data that would re-produce the motion of the OUT (See Figure 4).

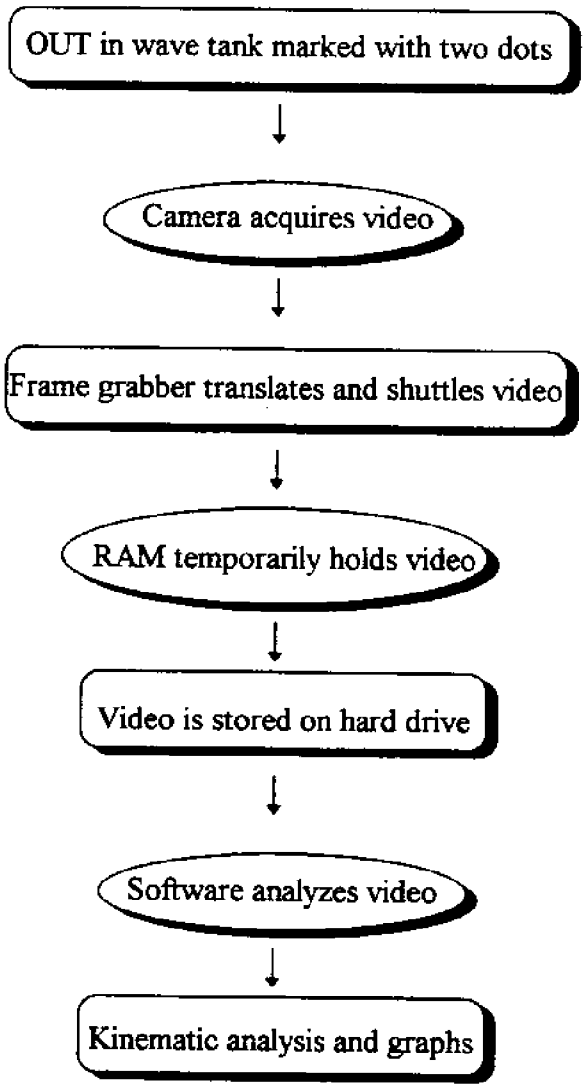


Figure 4: Flow Chart for Optical System

1. Testing Parameters

The primary considerations to ensure accurate and useful data were: sampling frequency, test length and data storage. If the sampling frequency is not chosen properly the reproduced signal will be erroneous. This phenomenon is known as aliasing and is prevented by selecting, the Nyquist frequency, a sampling frequency that is at least twice the maximum frequency of interest.

The maximum frequency of interest occurs when an OUT is being towed into waves. Assuming deep water wave characteristics, the encounter frequency for different wave periods a various model velocities is presented in Figure 5. The equations used to determine the encounter frequency are:

Deep water wave speed	$C_0 = gT/2\pi$
Deep water wave length	$L_0 = gT^2/2\pi$
Relative velocity	$V_{rel} = C_0 + V_{model}$
Encounter frequency	$F_{encounter} = V_{rel}/L_0$

Where T is the wave period, g is the gravitational constant and V_{model} is the velocity at which the model is being towed into the waves.

Encounter Frequency $F_{encounter} = (2\pi V_{model}/gT^2) + 1/T$ (1)

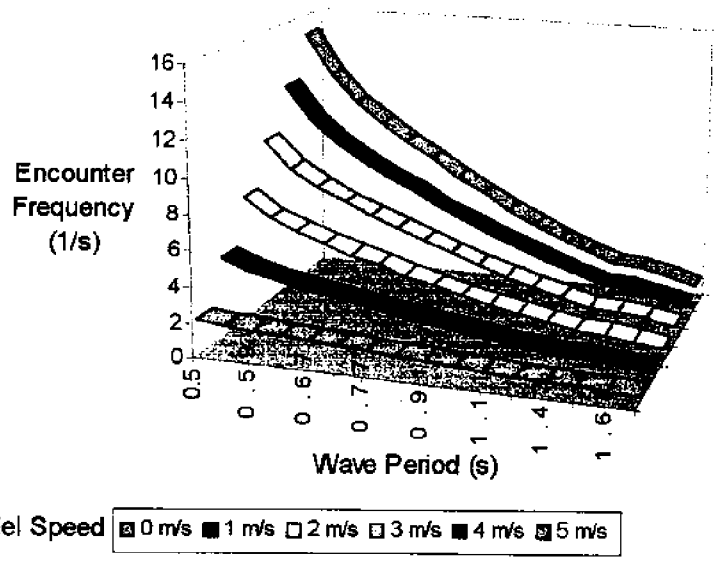


Figure 5: Encounter frequency versus wave period for a useful range of periods in the UNH/COE wave tank. Low wave period, high tow speeds produce the highest encounter frequency.

Once the encounter frequency was known, the sampling rate at which reliable data should be obtained can be calculated. The minimum sampling rate was determined by multiplying the encounter frequency by two, thus satisfying the Nyquist condition.

Minimum sampling rate $F_{sample,\ min} = (2)*(F_{encounter})$ (2)

The next consideration for accuracy is the camera's resolution which creates a limit on the minimum displacements that can be evaluated. For example, the resolution of a standard television is approximately 240 x 400. The first number, 240, represents the number of pixels the vertical axis is divided into; the horizontal axis is divided by 400. If an image represents a 1 meter by 1.66 meter area, each pixel would represent a 4.2 mm x 4.2 mm area (See Figure 6). As the resolution is increased, the area represented by a single pixel is decreased.

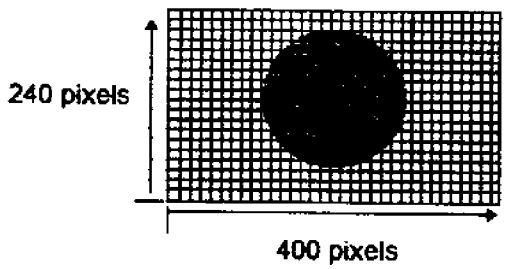


Figure 6: Pixel Division for 240 X 400 Resolution

If images are not transferred properly, data can be lost. An interface between the camera and computer was needed to transfer the images. Using image size and sampling frequency, the required transfer rate was found (See equation 3). This rate is an important consideration when selecting a frame grabber which is used to translate and shuttle the images.

Num. of pixels in one frame # pix = (Vertical Resolution)(Horizontal Resolution)

Transfer Rate $TR = (\#pix)(F_{sample})/(1*10^6)$ (Mbytes/s) (3)

Although the frame grabber may be capable of transferring the data to the computer, the computer must also be capable of storing the data at the same rate. RAM (random access memory) was the cheapest solution for storing the data as fast as it could be shuttled to the CPU. The amount of memory needed for a test was determined by:

Required memory Mbytes of RAM = (TR)(duration of test in seconds) (4)

where TR = Transfer Rate

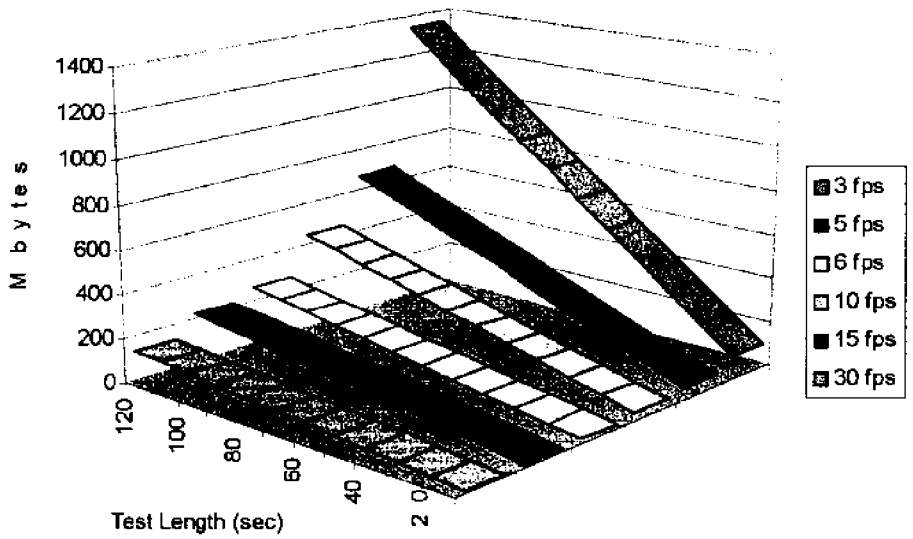


Figure 7 Mbytes of RAM required to run various test lengths at different sampling rates. High sampling rates, long test lengths produce the highest memory requirements.

After evaluation of the wave tank's capabilities, the overall system parameters were qualified and can be seen in Table 1.

Table 1: Parameters for Optical System	
Maximum Sampling Frequency	30 frames/sec
Minimum Resolution	640 by 480
Minimum Required Memory	256 Mbytes RAM
Minimum Transfer Rate	10 Mbytes/sec

2. Component Criteria

Having defined the criteria for the system, a comprehensive search for the components began. Each component was investigated thoroughly through interviews, trade shows, the WWW and professional contacts. Gradually, it became apparent that certain features would be required of the various components. The following information identifies the essential elements for each component which are essential for the success of the system.

CCD cameras

Charge Coupled Display, CCD, cameras consist of a solid state chip, as small as 1/3 of an inch square, on which the lens focuses the image. The solid state chip is divided up into individual elements instead of a continuous photosensitive surface, typical of traditional cameras. When each individual element is struck by light, it converts the light to a charge which is translated in the camera's circuitry to an analog or digital output.

The camera can read the image in various ways. An interlaced scanning camera initially reads every other line of the image to the end then returns to the beginning of the image and reads the remaining lines. If the process is not done quickly enough, a venetian blind effect can result where every other line is shifted by a few pixels. To circumvent this, a non-interlaced or a progressive scan camera can be used. Progressive scanning is a method of scanning image information off the chip in a line by line basis. With progressive scanning, full-frame images can be captured in very short durations.

The shutter speeds of a camera refer to the amount of charge that a chip absorbs during a set period. Images are electronically shuttered by scanning out only a portion of the charge that has accumulated over one field time.

Asynchronous reset allows the camera to reset and start a new scan when it is triggered or programmed to do so. This allows for the camera to capture and scan an image at specific time intervals permitting it to be triggered by an external source.

Frame Grabber

The frame grabber serves as a camera interface and acquisition device with the goal of moving images to the host (CPU) as quickly and efficiently as possible. Since the frame grabber is supporting the image transfer, the computer's RAM can be used for image storage, the CPU for image processing and the VGA card for image display. The division of these tasks allows for greater processing speed and a lower work load on the CPU.

Currently, there are frame grabbers on the market that have Direct Memory Access (DMA) Control. DMA is a technique of controlling data transfer at a continuous rate which improves upon the standard burst transfer rate found in older frame grabbers. There are two classifications of DMA control, conventional bus master and scatter/gather DMA. Conventional bus master DMA moves a single block of data to the host's memory. After the first block of data has been transferred, the controller chip interrupts the CPU to request permission for another block to be sent. This stop and go data transfer can create a 'traffic jam' in the DMA controller while it waits for a green light from the CPU. Scatter/gather DMA control allows continuous data throughput with minimal use of the CPU by establishing a series of DMA commands before interrupting the computer, freeing the CPU to perform other tasks. While the controller shuttles in the data, the scatter/gather DMA processor can continually add to its command series while the CPU is busy.

Frame grabbers can accept analog or digital camera output. VCR cameras and some CCD cameras provide non-standard analog output which requires an analog conversion module to be added to the frame grabber. Standard analog output (RS-170) is compatible with most frame grabbers. Analog signals carry various timing information multiplexed along with the data. This additional information can lower the number of gray scales by six percent, resulting in approximately 240 gray levels. More advanced cameras provide output in a digital format (RS-422) which may requires additional hardware. The digital output allows the full use of the 256 gray scales provided by the 8 bit digitization. Digital output also provides a higher signal to noise ratio resulting in more reliable data.

Some frame grabbers accept a color module which can allow the frame grabber to accept additional camera inputs. One color camera can be added or three monochrome cameras can be placed in each color input slot. A color module would permit the possibility of upgrading to three-dimensional analysis.

Output formats of frame grabbers vary. Most frame grabbers will output to typical image formats; i.e. BMP, TIFF, GIF, and TGA. These formats are merely different ways computer scientists have developed for storing images. Output format becomes a key issue when considering its compatibility with image processing software.

Computer

The CPU had be fast enough to handle all of the image processing applications in a reasonable time period. Real-time analysis, i.e. processing during data acquisition, was determined to be an unnecessary requirement. The computer had to have the ability to store data from the frame grabber at a rate of at least 10 Mbytes per second and have enough RAM to handle the taxing memory requirements brought on by image processing. Using equation four, it was approximated that a twenty second test at fifteen frames per second using a resolution (484 X 768) would use 110 MB RAM.

Because the video cannot be stored indefinitely in RAM, an archival system had to be created. A writable CD ROM provided a means to save both the full video sequence as well as the analysis.

Software

The image processing software had to be powerful enough to be meet the large range of tasks it might be called on to perform, yet it had to do so in a user friendly manner and provide useful output. A unique software system had to be created that quickly produced accurate and useful results. Numerous options for data analysis already existed on the market but held very high price tags. It was decided that the analysis program must be written by the group in order to assure proper data handling and application specific results at a reasonable price. When selecting the ideal software platform, the previous

programming knowledge of the group members had to be kept in mind. Suitable characteristics the software was required to have were expandability, easy to program, good memory management, and a user-friendly interface.

III. Acquiring the System

Having defined the basic criteria for each component, vendors were approached for pricing and product recommendations.

A. Entertaining Bids

The selection of CCD cameras that provided a sampling rate of 30 frames per second was limited in size, but not quality. Two companies, Pulnix and Sony controlled this portion of the market, providing relatively similar cameras in the same price range.

Every supplier that was approached found a different frame grabber that met all of the defined criteria. The options and features available were endless. A decision matrix (See Appendix E) was created to help evaluate the pros and cons of each frame grabber. Once the selection was narrowed down, vendors were approached for price quotations for both the cameras and frame grabbers.

After a full month of negotiations, JKN Electronics, Inc. in Bellingham, MA supplied the lowest bid. JKN offered the type of products that OPIE required, strong technical support and a 5% university discount. The specific products that JKN supported also aided in the final selection of the components.

B. Selected Components

The only camera which was found to have the necessary characteristics and still remained in a reasonable price range was the Pulnix TM-9701 (See Appendix D for specification sheet). This camera has upper end resolution (768 x 484) and can sample at a rate of 30 frames per second. The Pulnix is a progressive scanning monochrome camera that can deliver either digital or analog output. It can be triggered externally and has ten shutter speeds (infinity-1/16000 sec.).

MuTech's M-Vision 1000 (See Appendix D for specification sheet) frame grabber met all of the required parameters. The M-Vision 1000 is a scatter/gather DMA controlled frame grabber that is capable of acquiring both analog and digital video at 30 frames per second. The frame grabber can shuttle digital images into the computer up to a rate of 50 MHz and can assign various image formats. The M-Vision 1000 can accept up to four camera inputs providing the opportunity for 3-dimensional analysis. MuTech is located in Billerica, Massachusetts, and has provided strong product support.

Because of the stringent criteria the computer had to meet, it became apparent that an off the shelf system would not suffice. It was decided that the PC with all of the components deemed necessary to perform the its tasks would be built by a professional. A list of the required components was given to Joel Wright of InfoServe, who then built the computer.

OPIE's software needs were met by a program already licensed to the university, MatLab. This software program was originally created to perform advanced matrix manipulations, however, because of its ability to be customized, it quickly adapted for use in many fields. It now accepts software add-ons for performing a vast range of sophisticated tasks, the most pertinent to this project being the Image Processing Toolbox. This toolbox allows for the easy manipulation and display of digital images in MatLab. It accepts standard image types such as BMP and TIFF and allows the user to treat them as ordinary matrices.

Table 2: Summary of Purchased Components	
Component	**Reasons for Selection**
Pulnix 9701 CCD Camera	Sampling Rate of 30 frames per second
	Analog or Digital Output
	Monochrome
	484 X 768 Resolution
	Progressive Scan
	Asynchronous Reset
	Programmable Trigger and Shutter
MuTech MV-1000 Frame Grabber	Transfer Rates up to 50 Mbytes/sec
	Acquires at 30 frames/sec
	Scatter/Gather DMA Control
	Easy to Use Camera Interface Software
	Monochrome
	Compatible with Windows NT
	Upgradable to Color or Multiple Cameras
	Software Development Kit
InfoServe Computer	384 MB RAM (can hold up to 1 Gig)
	200 MHz Pentium-Pro
	4.2 Gig Wide SCSI Hard Drive
	Ultra Wide SCSI Controller
	PCI Bus
	Read/Write CD ROM
	MS Windows NT Workstation 4.0
Matlab Software	Image Processing Library
	Easy to Operate and Customize
	Partially Autonomous
	Provides Standard Data Output

IV. Image Analysis Software

A powerful analysis program had to be written that quickly analyzed video sequences and remained compatible with the video acquisition software. The challenges presented in the programming were locating the markers, calibrating the data and supplying meaningful output.

A. Marker Location

Once the components were obtained and assembled, the next task was to obtain position information from a video sequence. It was previously determined that the motion of the model could be traced by following the path of two markers. In order to do this, the exact location of the two points in each frame must be found. The easiest solution would be to manually select the two markers from each frame, creating a data log. However, selecting two points by hand for well over 500 frames is not the most efficient method. A process by which the computer automatically locates the markers was developed.

When images are loaded into MatLab, they are converted into a numerical matrix, each element of the matrix contains a gray scale value; a zero value represents a pure black pixel while a 256 value represents white. By locating the lowest value in the matrix, the darkest point in the scanned area is found.

The search is begun by looking down each column within the search area. Every time the computer finds a gray scale value lower than any other it has come across, it records both the new minimum gray scale value and the row and column of the value. When the search is finished the row and column of the darkest pixel are saved as the position of the marker.

Whenever the gray scale value is equal to the minimum value found previously, the computer ignores it and continues on to look for darker pixels. Thus, if two pixels in a search area are of the same intensity, the first pixel located will be identified as the marker. By consistently selecting the first pixel found, the darkest, upper left pixel of a marker will be the pixel that is traced throughout the testing sequence.

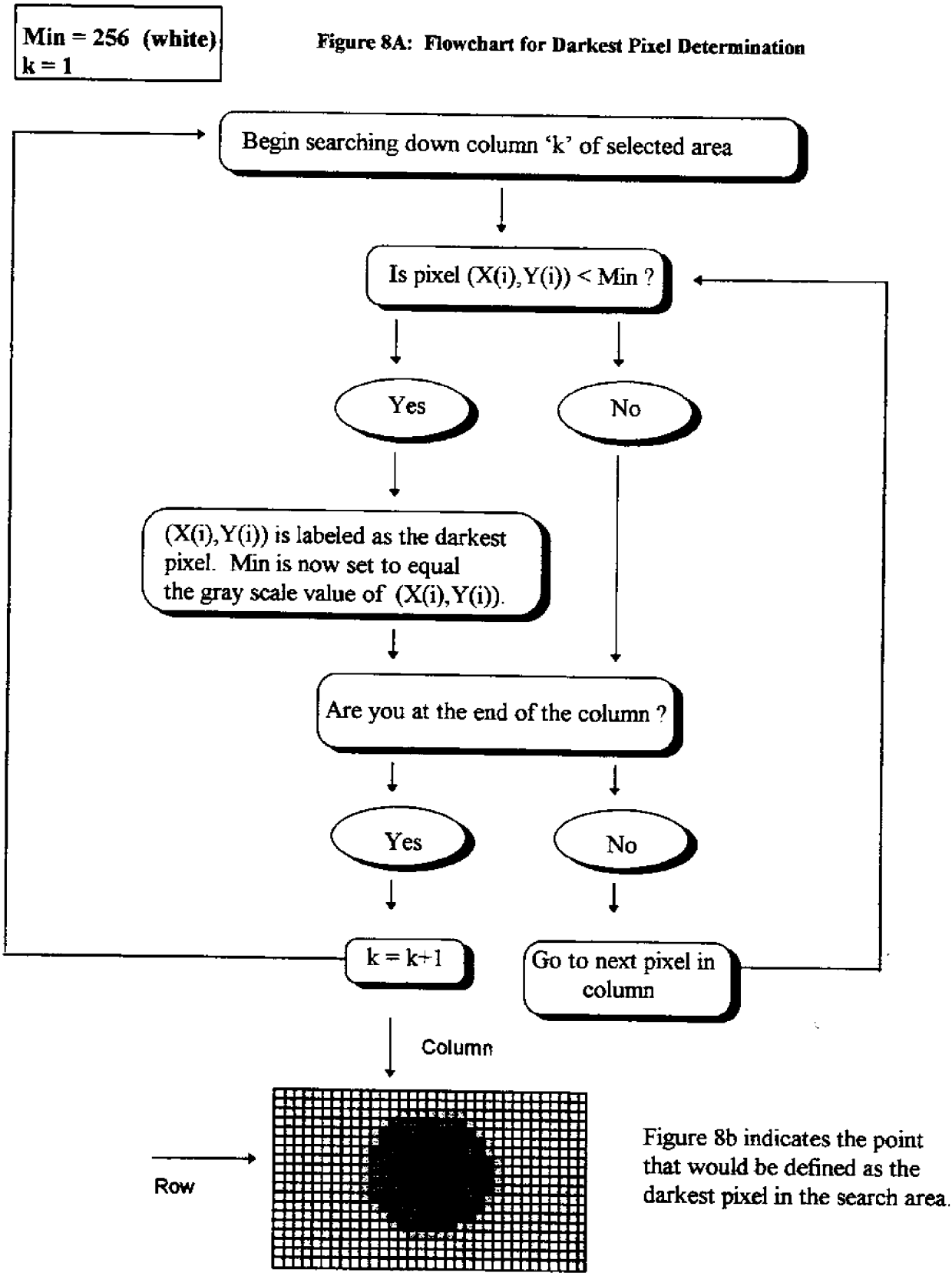


Figure 8A: Flowchart for Darkest Pixel Determination

Figure 8b indicates the point that would be defined as the darkest pixel in the search area.

Three methods of defining an area to search for the marker position were evaluated: area scanning, Taylor series prediction and last point prediction.

1. Area Scanning

Area scanning began by defining an area of interest which is then divided in half vertically. Each half of the image was scanned independently to locate the lowest gray scale value. (See Figure 9).

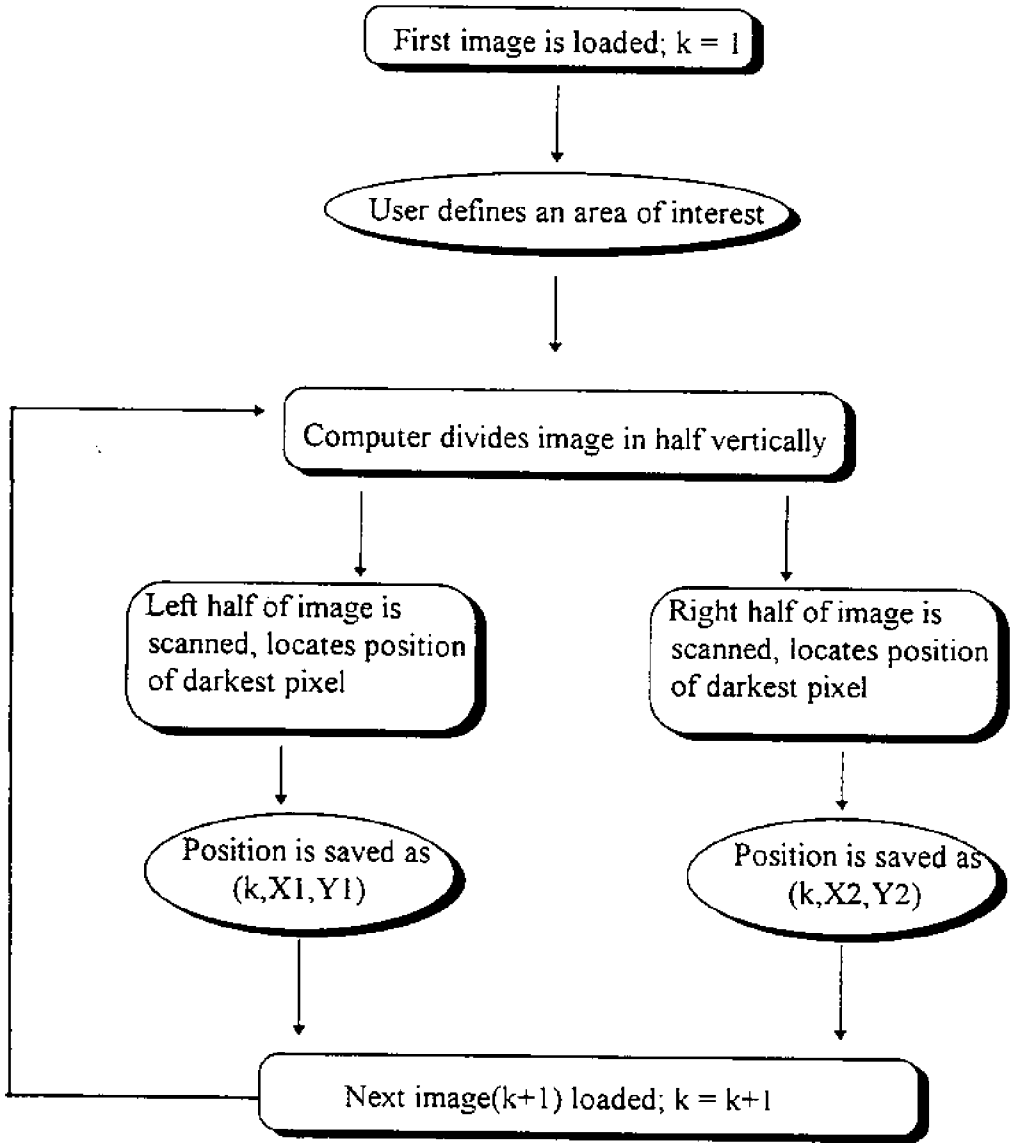
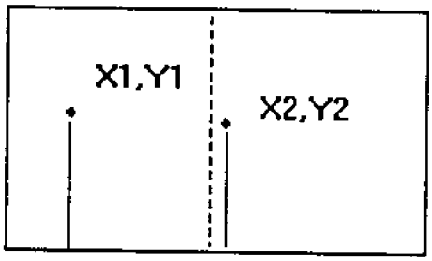
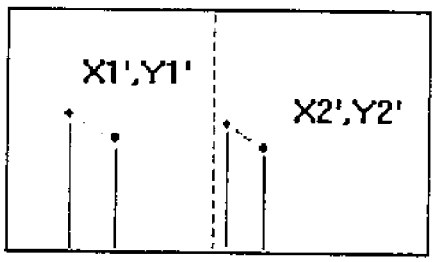


Figure 9: Flow Chart for Area Scan Locator

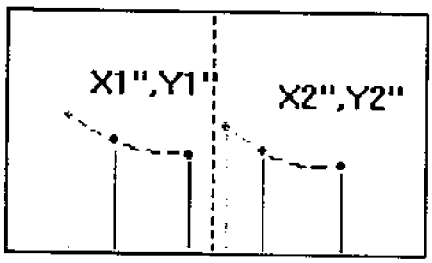
This process presented two problems. First, it was difficult to define an area of interest for all of the frames because the maximum displacements were not known. Second, the area scan method failed whenever both dots entered the same half of the area (See Figure 10). When this occurred, one half would find the darker of the two points and the other half would find a random dark spot, typically a shadow.



t[x]



t[x+1]



t[x+2]

Figure 10 depicts the motion of two points for three time steps. It can be observed from these images that at t(x+3), both markers will exist in the same half-plane. When this occurs, inaccurate data is generated and the Area Scan method is of little use.

2. Taylor Series Prediction

The complications developed using a static search area were circumvented by predicting a search area that moved in with markers. Using the Taylor series, a prediction method was formulated that estimated the next position of a marker. An area around that predicted point was then searched (See Figure 11). The analysis began with the user defining the position of the two markers for the first three frames. This was necessary because the prediction requires the first three locations to estimate the location of the fourth position. Assuming planar, rigid body motion, the distance between the points (D) will never change. Therefore, by defining search circle of radius D or less, it would be impossible for two points to exist in the same search area. The computer centers the search circle on the

predicted position (See Figure 12). The darkest pixel was found in the same manner as in the area scan.

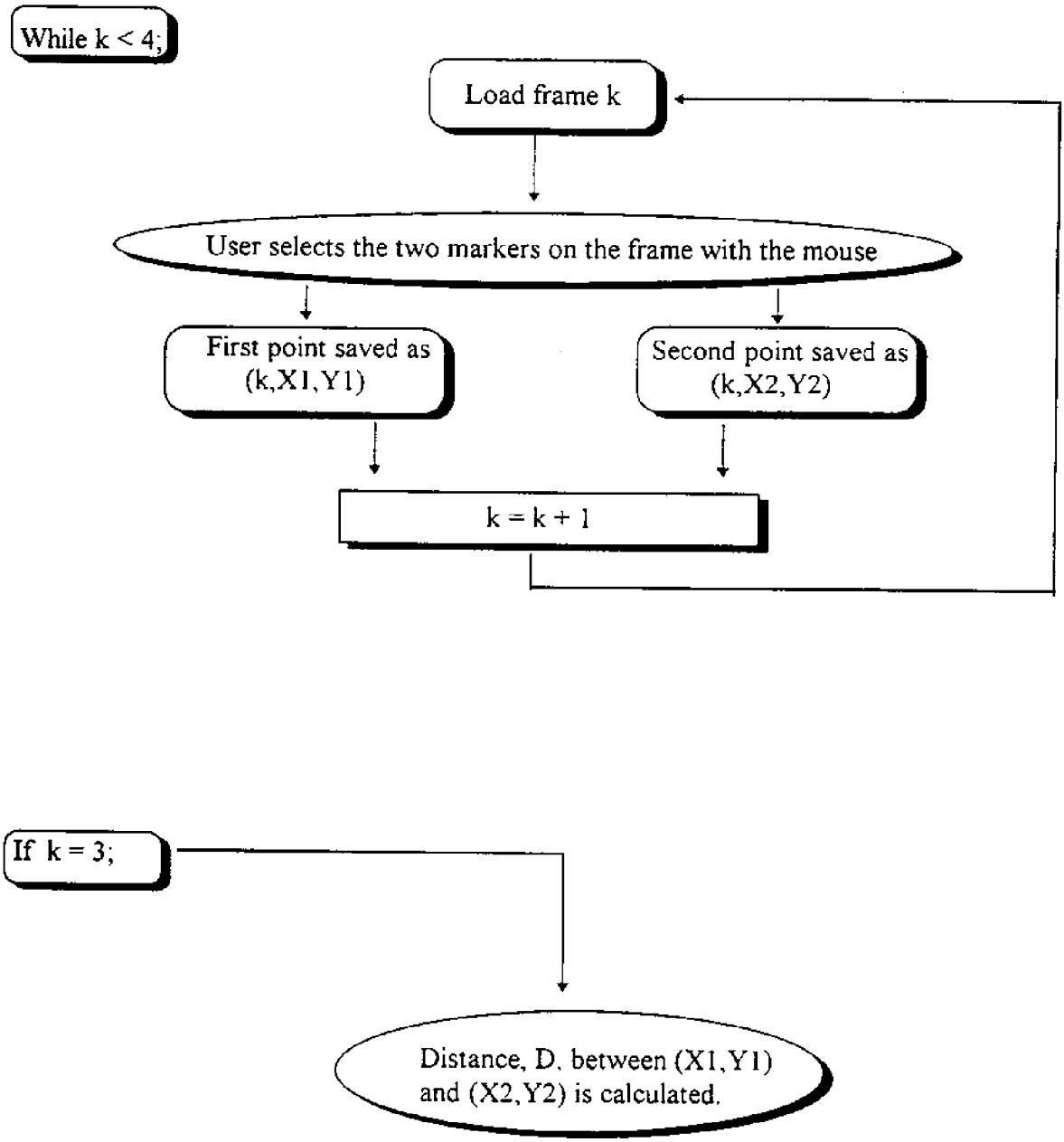


Figure 11a: Flow Chart for Taylor Series Prediction

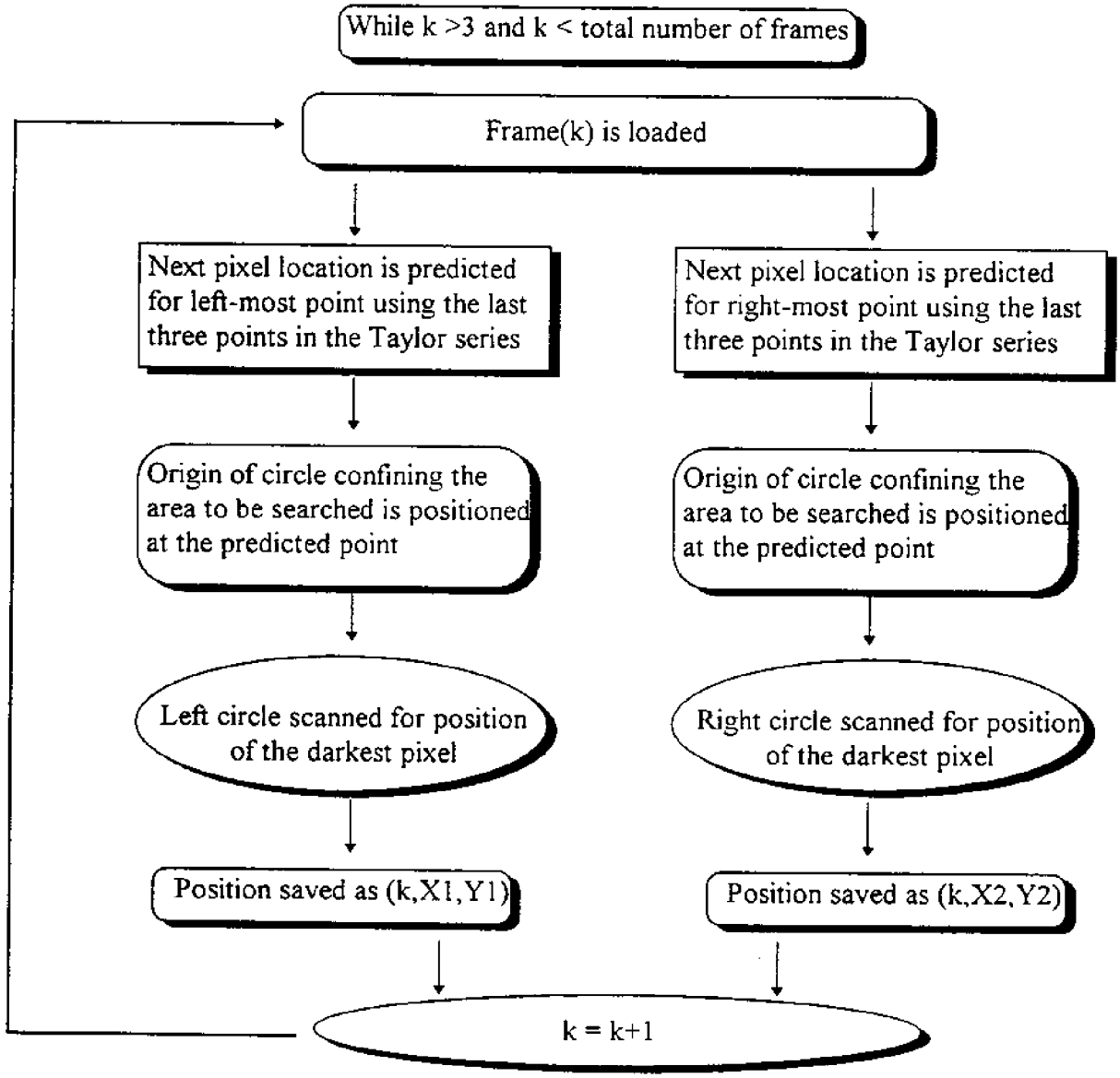


Figure 11b: Flow Chart for Taylor Series Prediction Continued

The Taylor Series approximates a function about a point, c, by differentiating and integrating each term in the series.

$$\text{Taylor Series} \quad f(x) = f(c) + f'(c)\,(x-c)/1! + f''(c)\,(x-c)^2/2! + \ldots \qquad (5)$$

Because the true derivatives at point c were not known, they had to be approximated through the use of a difference technique (See equation 6). Equation 6 was used to predict the x-position; the y-position was predicted in the same manner.

$$\text{Predicted Position} \quad x_{n+1} = x_n + \Delta x + (\Delta(\Delta x))/2 \qquad (4)$$

$$= x_n + (x_n - x_{n-1}) + 0.5((x_n - x_{n-1}) - (x_{n-1} - x_{n-2}))$$

$$x_{n+1} = 2.5x_n - 2x_{n-1} + 0.5x_{n-2}$$

After this prediction locates the marker, the series is updated by incorporating the new point and repeats this routine for each frame. This process had difficulties locating the correct points when sampling near the Nyquist frequency.

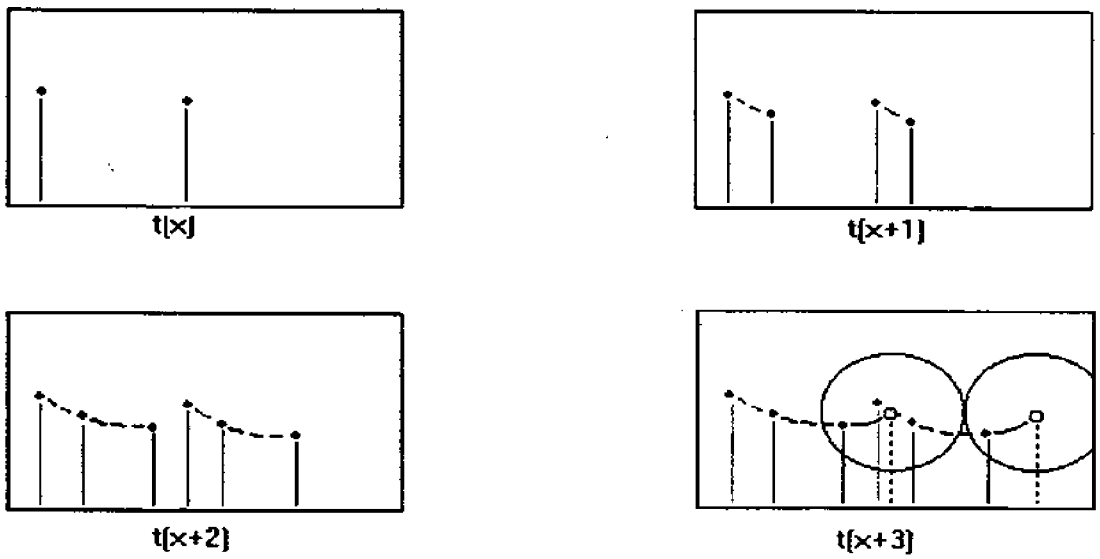


Figure 12 depicts the motion of the two markers for three time intervals. The Taylor Series prediction tells the computer where to locate the 'search circles' as seen at t(x+3).

3. Last Point Prediction

The final method, last point prediction (LPP) allows for accurate sampling near the Nyquist frequency. The last point prediction (LPP) centers the next search area on the last known location of the marker. The search area was defined by the same process used in the Taylor series prediction and is of the same size (a circle of diameter, D). This method

works when the marker doesn't move out of the search area in the time between two frames (See Figure 13).

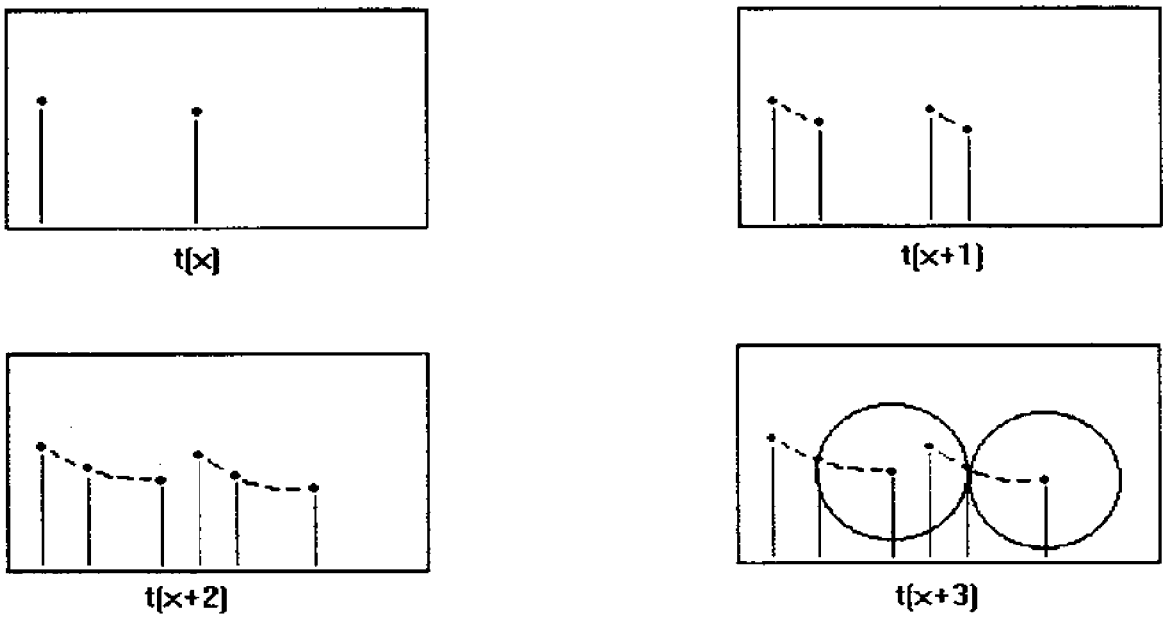


Figure 13: LPP at various time intervals.

LPP worked better than Taylor series prediction because the derivative approximation in the Taylor series was inaccurate at low sampling rates. This caused the search area to start in the wrong place and consequently the darkest pixel found was not the marker.

The maximum velocity that can be measured with LPP can be found by:

$$V_{max} = (D)\ (F_{sample}) \qquad (5)$$

If the maximum velocity of the model is known, equation 5 aids in the determination of the sampling frequency as well as the distance between the markers.

Inaccuracies can occur using LPP when the marker separation is less than the distance they travel between frames. In testing situations when this occurs, the Taylor series prediction is the better method to use.

B. Calibration

In order for the position information to be of any use, the output had to be converted from pixel position to a conventional units system. This was accomplished by determining two calibration factors, one for each axis. The two factors were essential because the camera pixels are rectangular, the horizontal direction being the larger of the two.

The calibration was performed by obtaining an image of a dark circle of known size. The user had to position the circle at the same distance from the camera as the model. The calibration had to be done only once, therefore, the calibration circle didn't need to be in the testing video. After the location of the circle had been defined by the user, the computer located any points in the area darker than a user-defined intensity. The user was shown a picture of the area with all of the filtered pixels turned completely black and markers indicating the row and column with

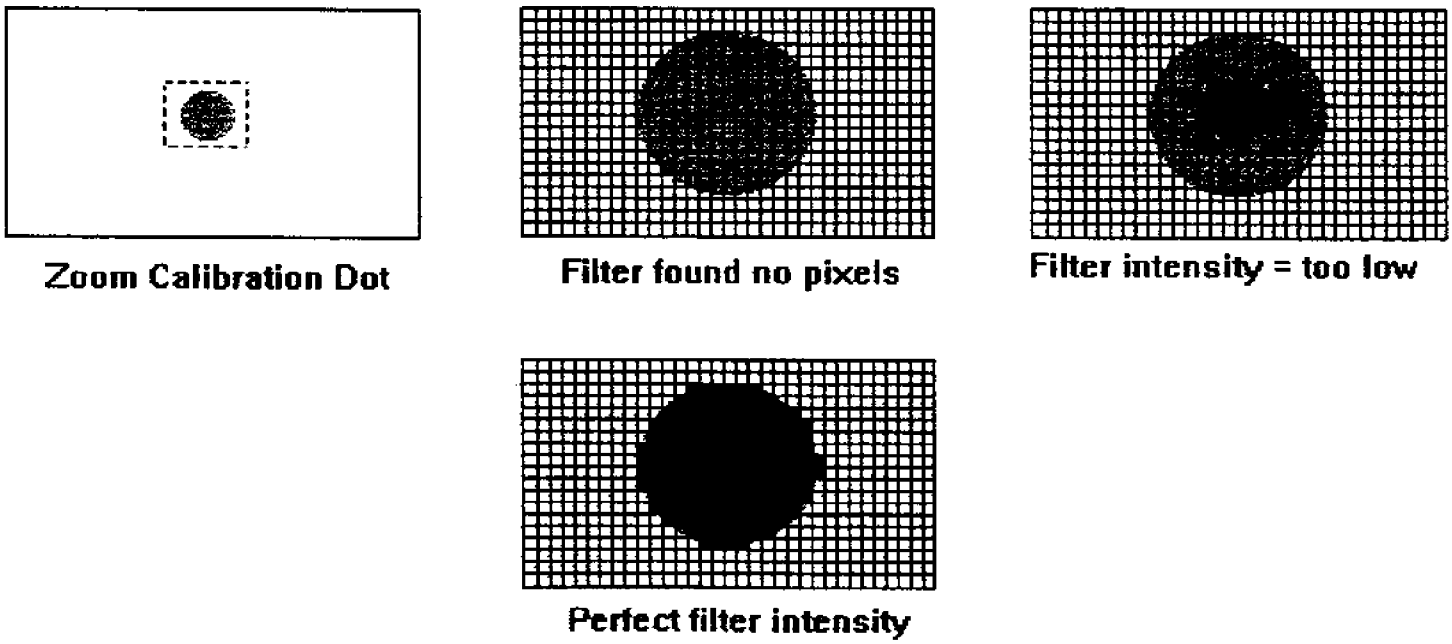


Figure 14: Calibration Filters

the most filtered pixels. The row and column that the markers indicate ideally crossed at the center of the circle. The user was given the option to redefine the threshold level if they were not satisfied with the results. The true diameter of the circle was divided by the horizontal and vertical maximums resulting in the calibration factors (See figure 14).

Calibration factor $calx = D / (\#\text{ of filtered pixels in horizontal direction})$ (6)

where D = true diameter of the circle

C. Frame of Reference

Defining a frame of reference for each marker was necessary to obtain meaningful displacement information. The user must select equilibrium positions for each marker by manually choosing them from one static frame. The reference marks define an origin for further calculations.

D. Data Analysis

Using the displacement of each marker from their defined reference points, it was possible to calculate the velocities and accelerations of the points. The velocity at a certain time step was found by dividing the change in displacement by the time between frames (See Equation 7). The acceleration was found using a similar technique except that used the change in velocity (See Equation 8).

Sampling period $T = 1/F_{sample}$

Velocity $V_{x,n} = \Delta x / T$

$V_{x,n} = (x_n - x_{n-1})F_{sample}$ (7)

Acceleration $accel_{x,n} = (V_{x,n} - V_{x,n-1})F_{sample}$ (8)

Angular displacement, angular velocity and angular acceleration are found in a similar manner. The angle between the two markers was found by calculating the arcsine of the separation of the two dots (Δy or Δx) divided by the total distance between them (D). However, if the angle was between 90° and 360° degrees a multiple of $\pi/2$ needed to be added to provide the actual angle (See equation 9). Through this method an equilibrium angle and angular displacement may be found. The angular velocity is found by dividing the change in angle between frames by the time between frames and the angular acceleration by dividing the change in angular velocity by the time between frames.

Angle measurement

in the first quadrant	$\theta = \text{Arcsine}(\Delta y/D)$	(9a)
in the second quadrant	$\theta = \text{Arcsine}(\Delta x/D) + \pi/2$	(9b)
in the third quadrant	$\theta = \text{Arcsine}(\Delta y/D) + \pi$	(9c)
in the fourth quadrant	$\theta = \text{Arcsine}(\Delta x/D) + 3\pi/2$	(9d)

Angular displacement $\theta_{dis} = (\theta_n - \theta_{ref})F_{sample}$ (10)

Angular velocity $\omega = (\Delta\theta_{dis})F_{sample}$ (11)

Angular acceleration $\alpha = (\Delta\omega)F_{sample}$ (12)

IV. Testing and Evaluation

Bench trials were conducted to test the accuracy of the computer code. These trials provided data that was useful in refining the experimental design.

A. Pendulum Test

The first test consisted of a white piece of wood with two small black painted dots suspended by two strings which permitted the wood to move in a pendulum type motion (See Figure 15). The purpose of this test was to evaluate the method of finding the dots and calibrating the data. Initial trials using the area scan indicated that the method was inaccurate whenever the dots moved into the same half of the area of interest.

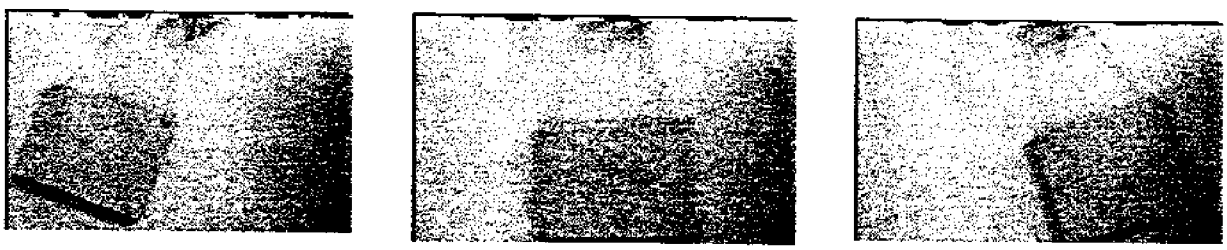


Figure 15. Bench Testing of Pendulum Motion at Various Time Periods

This result initiated the exploration of prediction methods. Both prediction methods worked much better than the area scan, however, the Taylor series prediction failed on sampling rates at which the LPP continued to work. These tests finalized the method of finding the markers and refined a method for calibrating the data.

B. Spinning Disk Test

The second test that was performed used a white disk attached to an electric motor (See Figure 16). This test helped to define the frequency at which sampling should be done and enabled the calculated velocities, accelerations, angles, angular velocity and acceleration to be compared to known values.

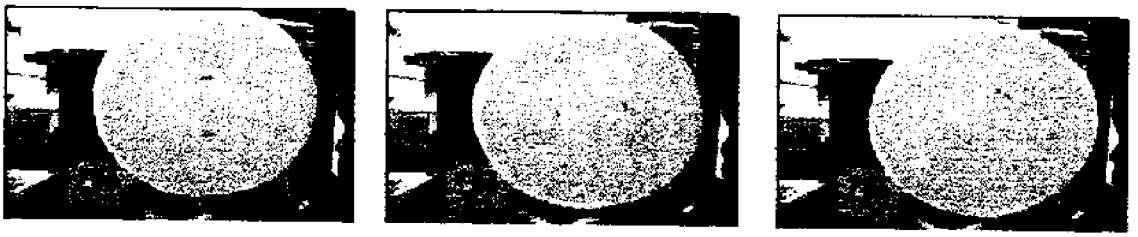


Figure 16: Spinning Test at Different Time Periods

The disk-motor setup was equipped with a cycle counter (incremented one value for each revolution). Using this counter over a period of time it was possible to calculate the angular velocity which was kept constant during the test. Knowing the angular velocity (ω) and the angular acceleration ($\alpha = 0$) it was possible to calculate the velocities and accelerations which occurred at certain points. The results from the spinning disk test can be viewed in Appendix (B)

While the data obtained from the system resembled the actual events taking place during the test, the output was noisy. However this was probably not due to inaccuracies in the system as much as it was due to older testing equipment.

C. Fish Cage Test

The laboratory tests provided information about the best data analysis approach; the LPP method was clearly the best approach. Knowing this, a real test was performed on a simple structure in the wave tank. Before this test was run, it was unsure whether or not glare on the tank window, shadows, spots in the back ground or the water's surface would interfere with the experiment. The test analyzed fish cage motion represented by a simple cube model (See Figure 17) The model was subjected to many wave spectrums to get an overview of its frequency response.

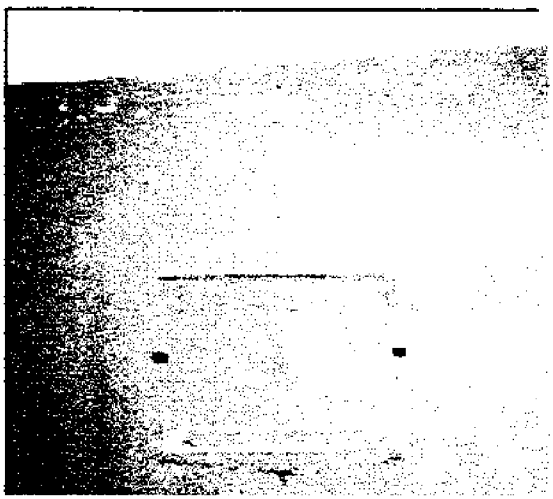


Figure 17: Fish Cage Model in Wave Tank

The prediction system had no problems tracking the points for any of the tests. (See results in Appendix B) Shadows, glare and the back ground had no effect on the test. The only problem appeared in the tendency of the model to twist which presented the camera with an oblique view. While this tendency did not hinder the system in finding the marker position in this test, it did challenge the initial assumption of planar motion. A method is needed to constrain the model to planar motion while having a minimal influence on the object's natural motioned. The constraint should be prepared for each test individually.

VI. Budget

Upon initial investigation of the current optical positioning systems, it appeared that a pre-assembled turn key system would cost approximately $20,000. Knowing that this figure was beyond the financial capacity of OPIE's budget, it was decided that the system would be created from components. Table 3 outlines the expected and actual budget for the entire OPIE. system.

Table 3: Budget for OPIE

Component	Estimated Cost	Actual Cost
Frame Grabber	$3,000	$1,354
CCD Camera	$3,000	$2,195
Lens	$500	$110
Power Supply & Cables	$500	$206
Image Processing Software	$2,500	$0
Computer	$6,500	$8,600
Additional Software	$500	$908
Tripod	$100	$89
Miscellaneous	$150	$80
Total	$16,770	$13,552

VII. Summary

The OPIE analysis system was created to measure the planar motion of various objects placed in the wave tank. The user-friendly software package specifically designed for OPIE provides the user with quick, accurate analysis of video giving velocity and acceleration information that can aid in research and structure design. The entire system is flexible and can be easily applied to many applications. Every component contains the latest technology as well as provides the potential for future upgrades. Initial tests have demonstrated the system's reliability in a broad range of applications. Tests are already scheduled to further research in oil spill containment technology and OPIE shows great promise for continued academic and financial returns.

VII. References

Figliola and Beasley, Theory and Design for Mechanical Measurements, 2nd Edition, John Wiley and Sons, New York, Copywrite1995

MuTech Corp., MV-1000 Grab Sequence Application Ver. 1.3 for NT, Copyright 1997

Stewart, James, Multivariable Calculus, 2nd Edition, Brooks/Cole Publishing, California, Copyright 1991

The Math Works Inc., Matlab for Windows, Copyright 1994

Appendix A: OPIE USER'S GUIDE

Software User's Guide

Version 1.0

WELCOME!

OPIE is a unique software system that allows a user to quickly acquire, save and analyze video. Before capturing data, one should conduct preliminary steps to insure accurate, high quality, memory efficient video sequences.

I. Acquiring Video Sequences

After all of the testing parameters have been determined, video acquisition can begin. ***MV-1000 SEQ*** is a data acquisition program obtained from MuTech that channels the input from the camera to the computer. To activate the program, select the icon: 

from the task bar. The following window will appear:

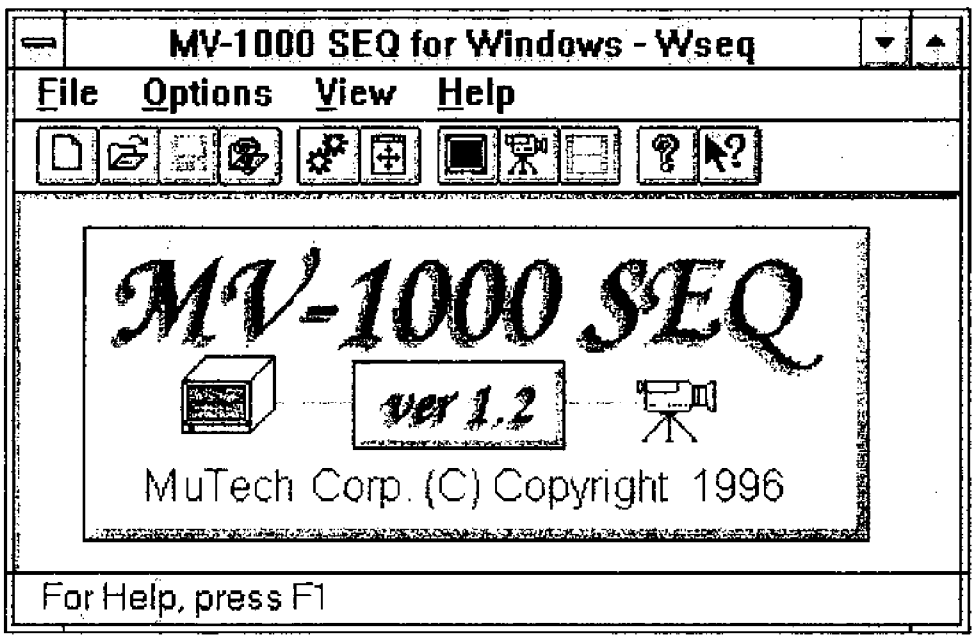
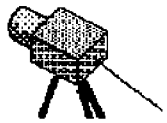


Figure 1: WSEQ Main Menu

A. Set Camera Configuration



1. Select "*Options*" from WSEQ's menu bar.
2. Select "*Select Camera*"
3. Select "*Non-Standard*"
4. Change directory to: "*c:\MV1000\CAmCfg*"; select file "*Tm9701a.ini*"

B. Define Test Parameters

1. Select "*Options*" from WSEQ's menu bar.
2. Select "Set *Record Options*"

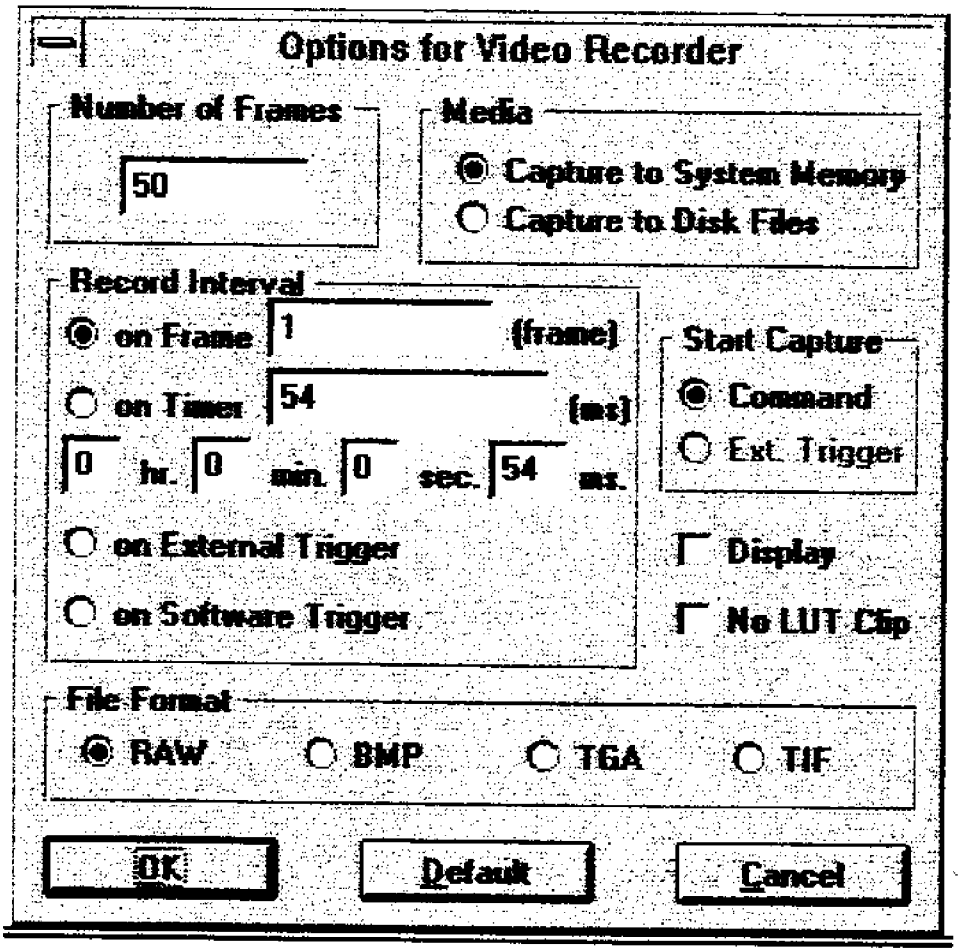


Figure 2: Record Options

3. Set:
   a. Record Interval "*On Frame...*"
      The Pulnix 9701 captures at a constant 30 frames per second. To capture at 15 frames per second, change "On Frame" to 2. For 10 frames per second, on frame 3, etc. The timer option allows for non-standard timing intervals.

b. Number of frames = (length of test (sec)) X (frames per second)
c. File format = *BMP*
d. Media: "*Capture to System Memory*"
e. *Disable the Display* if capturing at a rate higher than 20 frames per second. (The display option allows for you to view the data while you are capturing. At high transfer rates, the computer cannot save and display the images simultaneously.

C. Preview Image Area

1. By selecting the "Adjust Video" Icon, a live picture is generated of the test area. The aperture and focus can then be adjusted accordingly.

D. Capture Video

1. When you are ready to record, select the "*Capture Video*" icon.
2. Select the 'play' button on the recorder and data acquisition will begin.

E. Save to Disk

1. Select "*File*"
2. "*Save As*"
3. Open folder: "*c:\MATLAB\BIN\VIDEO*", enter a file name with a maximum of **four** characters.
4. "*Save*"

II. Analyzing Video Sequences

All image analysis is done through MATLAB with OPIE.
To start the OPIE program, click on the OPIE icon.

1. The opening screen will appear (See Figure 3)
   a. Select 'Start Analysis'
2. 'Testing Parameters' screen will appear (See Figure 4)
   a. Enter information
   b. Select 'Continue'

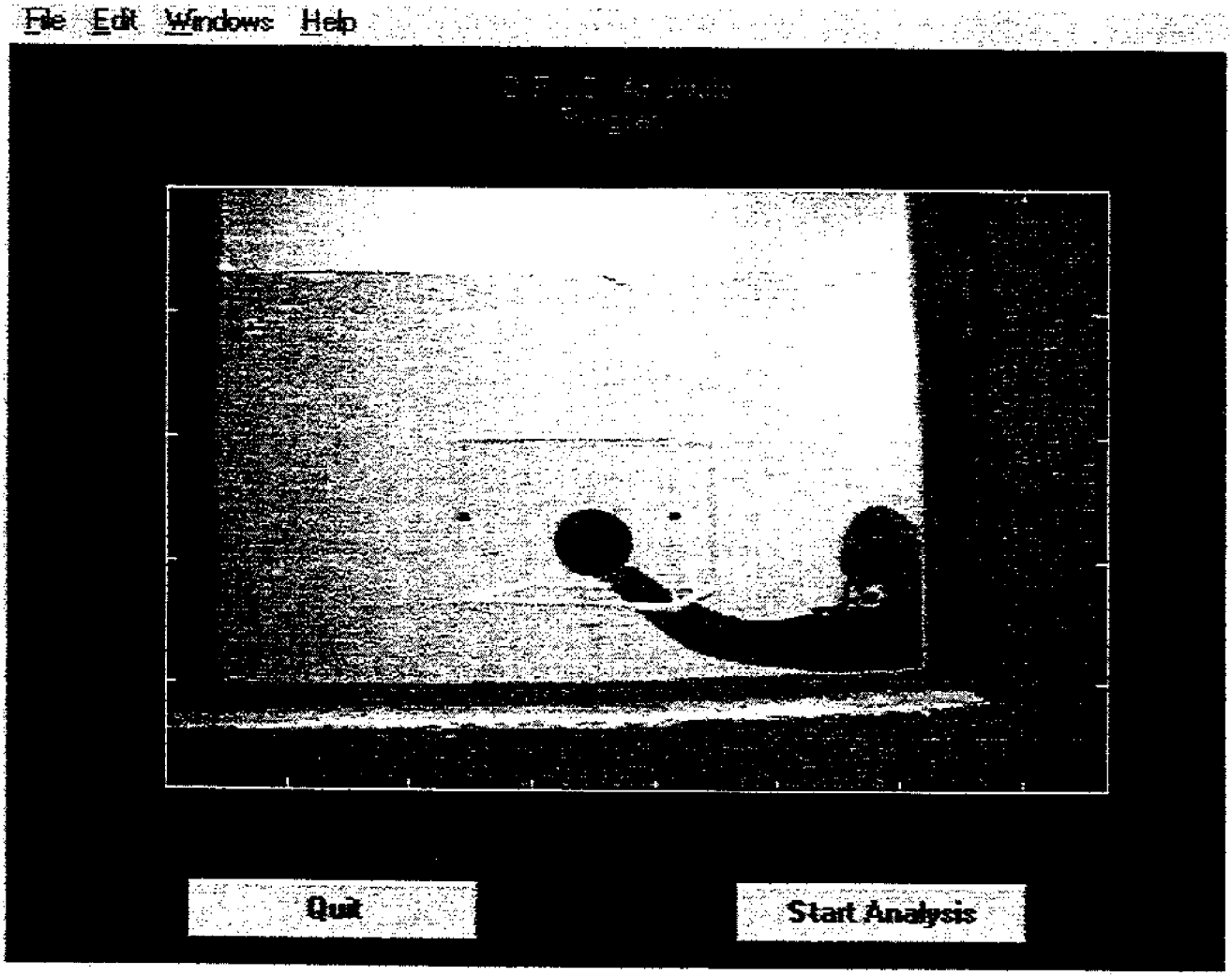


Figure 3: OPIE Program Starting Frame

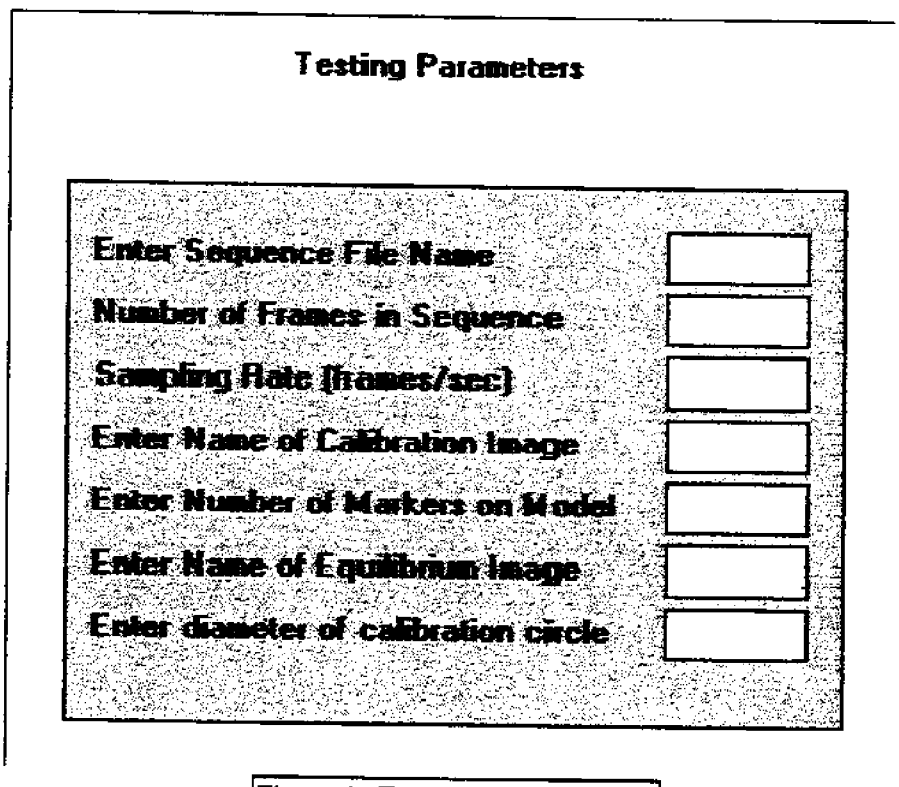


Figure 4: Testing Parameters

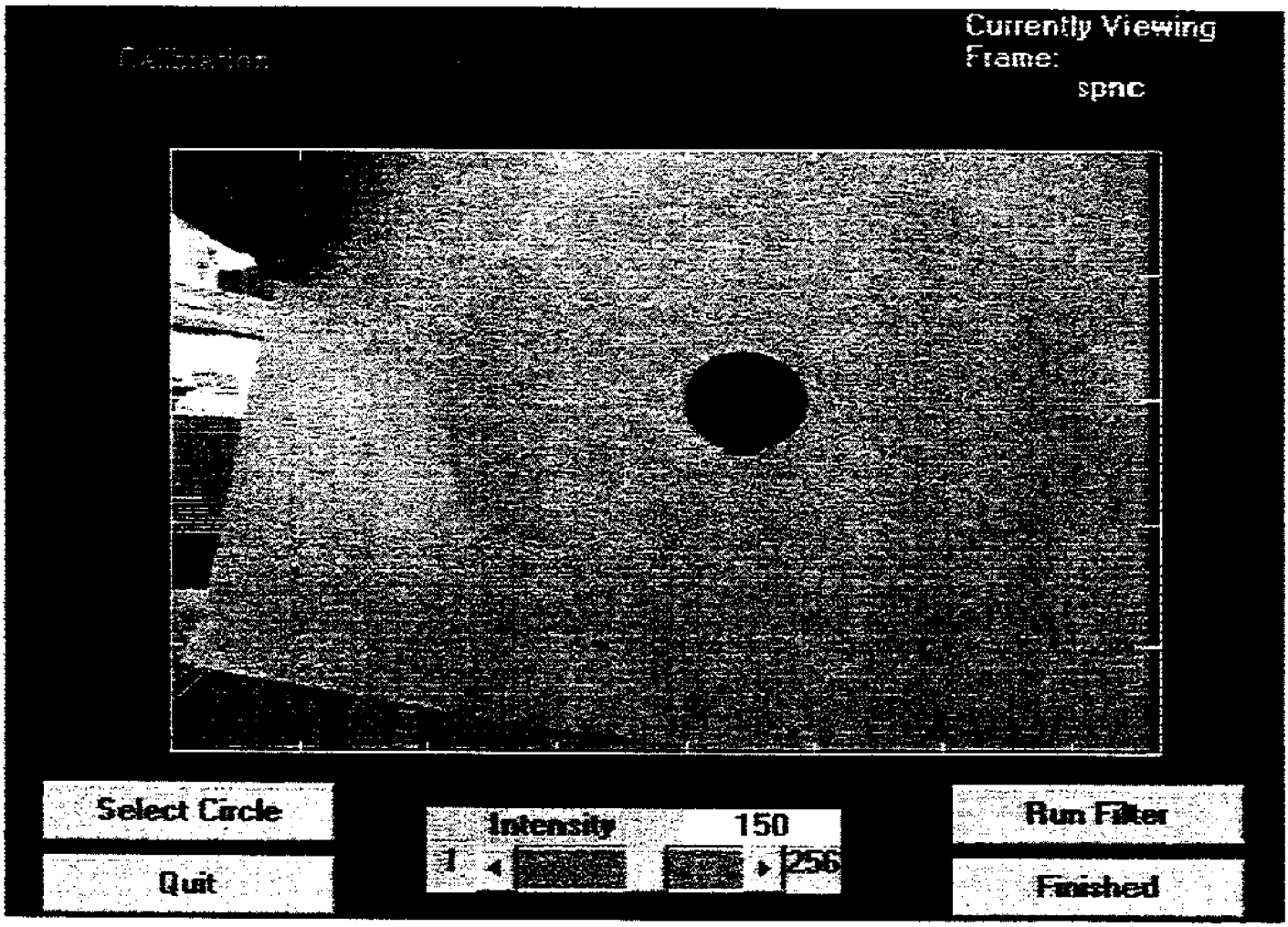


Figure 4: Calibration Screen

3. Calibration

   a. Click '*Select Circle*'
   b. Drag mouse to create a square which encompasses the calibration circle. The image window will zoom to the selected area.
   c. Set intensity level. (256 = white, 1 = black)
   d. Click '*Run Filter*'

      The area selected in step 3A will now be filtered. The desired filter is at an intensity level which results in the filter pixels covering the entire calibration circle.
   e. If you are not satisfied with the results, you can repeat the process by re-selecting a new intensity level, then clicking '*Run Filter*'
   f. Once satisfied click '*Finished*'

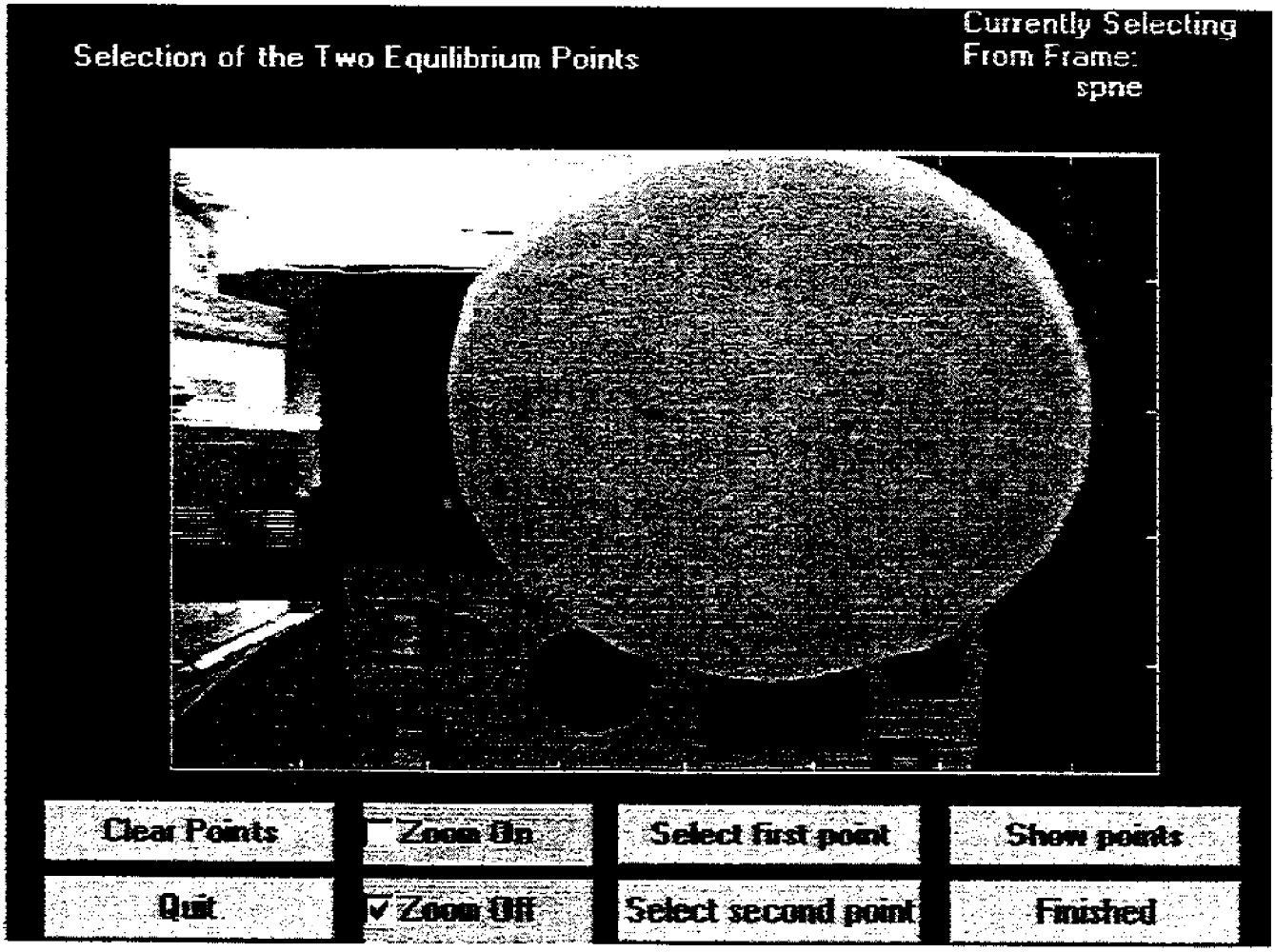


Figure 5: Reference Point Input Screen

4. Reference Points
    a. To Zoom:
        1. Active Zoom by clicking check box
        2. Either drag the mouse over an are to zoom, or simply continuously click on the area you wish to enlarge. To zoom out; click the right mouse button.
    b. To Select First Point:
        1. Click '*Select first point*'
        2. Use mouse to select point
    c. To Select Second Point:
        1. Click '*Select second point*'
        2. Use mouse to select point
    d. To View Selected Points:
        1. Click '*Show Points*'
    e. If you are satisfied with the selected points, click '*Finished*'. Else click '*Clear Points*' and repeat process.

5. Predictor Points (*Screen format is the same as Figure 5*)
    A. Repeat the same process for selecting points as done in step 6.
    B. Select '*Finished*'.

Once 'Finished' is selected, OPIE will evaluate the entire frame sequence (the number of frames to be analyzed is inputed by the user). As each frame is analyzed, it appears in the active figure window. Once the program identifies where the two markers are located, it will draw cross-hairs indicating their position. After all of the frames have been analyzed, the graphs for position, velocity and acceleration appear automatically. A matrix of the data is also available for the user.

Appendix B: Initial Test Results

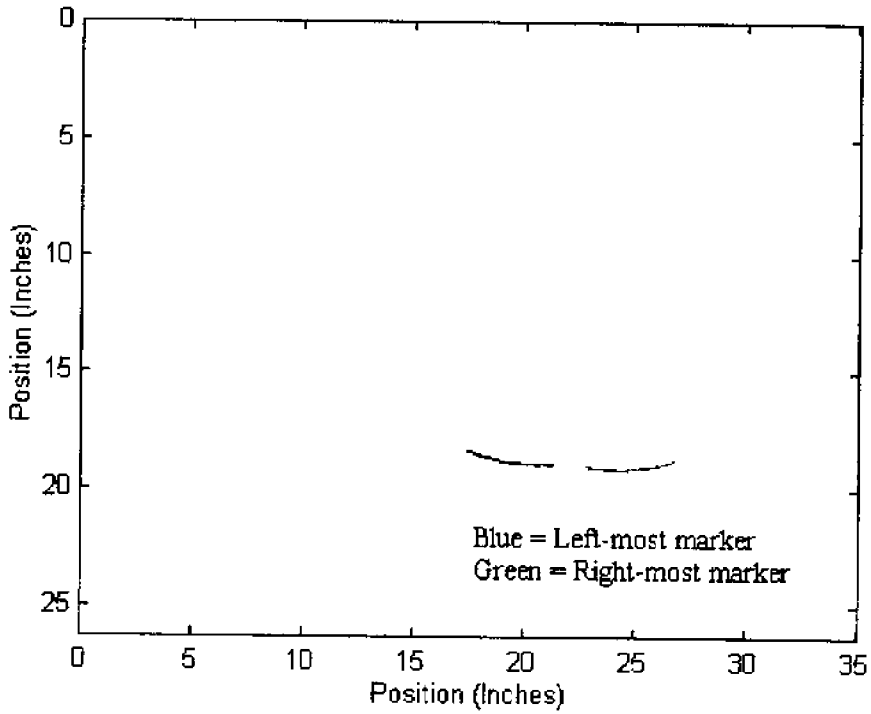


Figure 1: Position of the Markers
Test: Pendulum

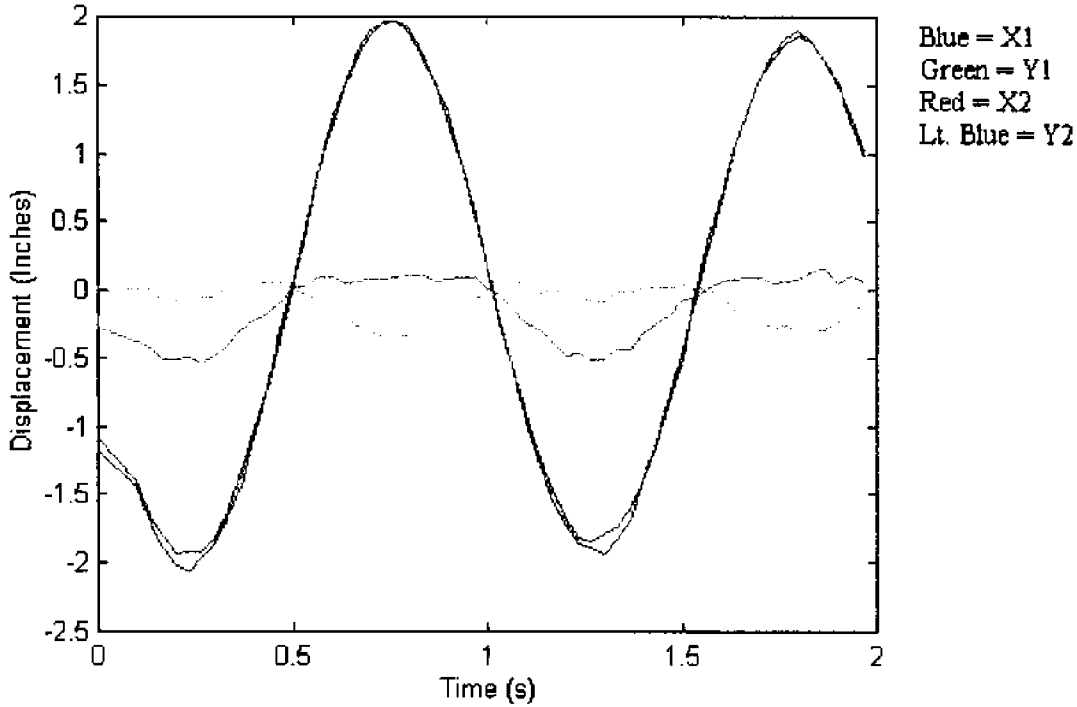


Figure 2: Horizontal and Vertical Displacement from Equilibrium
Test: Pendulum

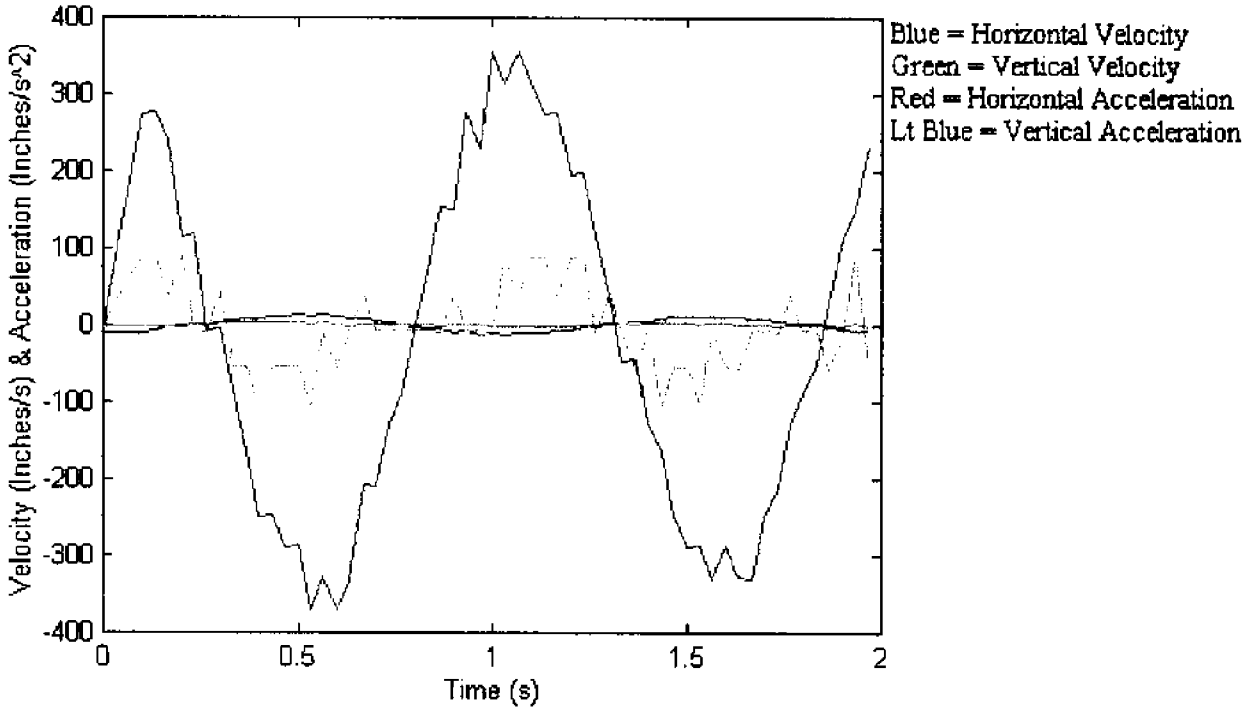


Figure 3: Velocity and Acceleration Components of the First Point Test: Pendulum

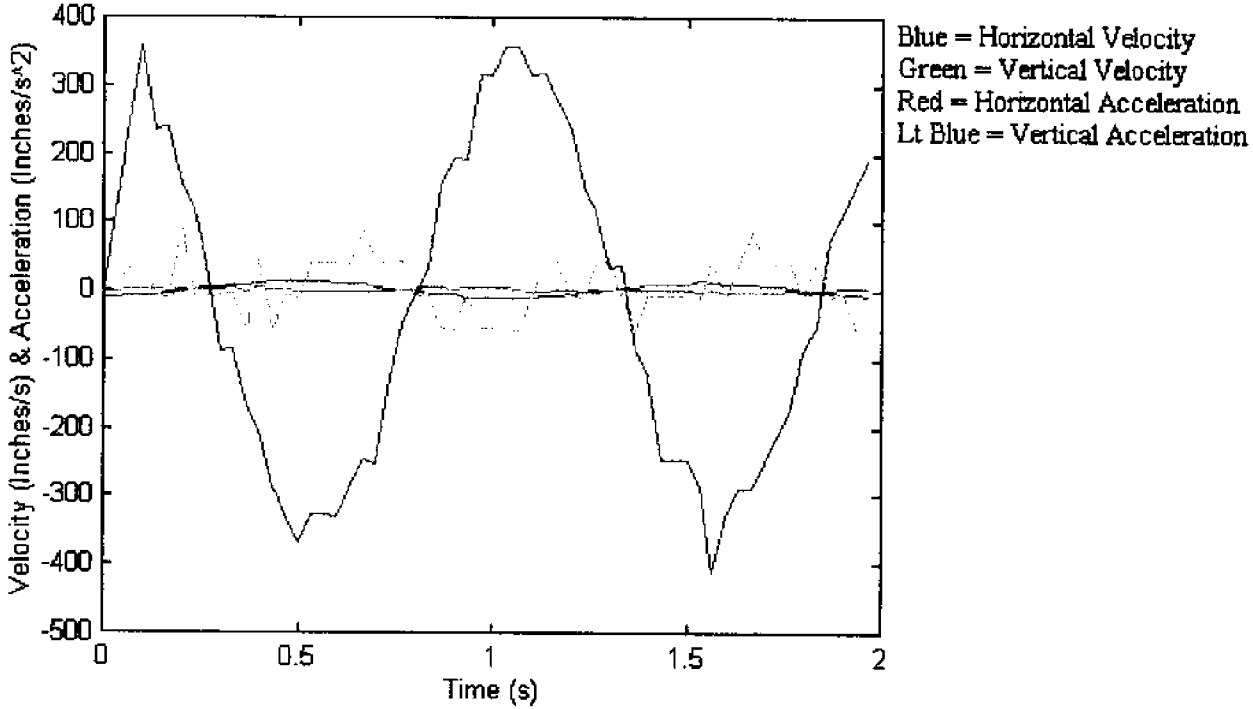


Figure 4: Horizontal and Vertical Velocities and Accelerations for the Second Point
Test: Pendulum

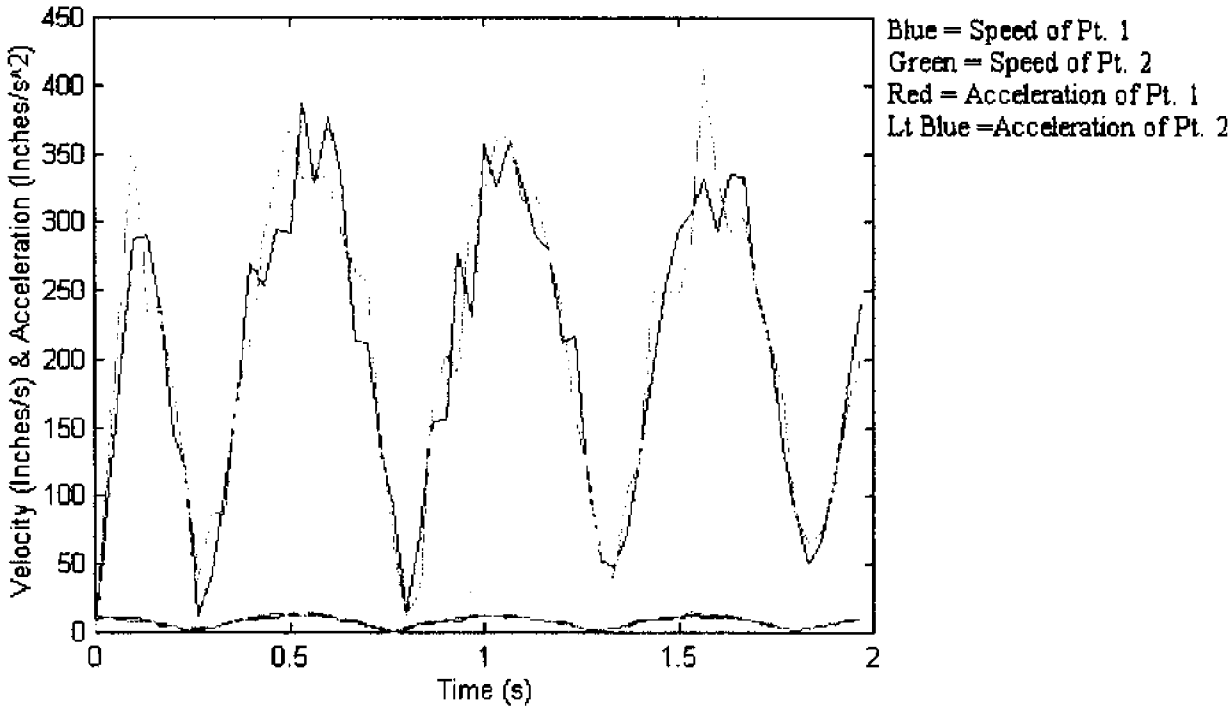


Figure 5: Speed and Total Acceleration of Both Points
Test: Pendulum

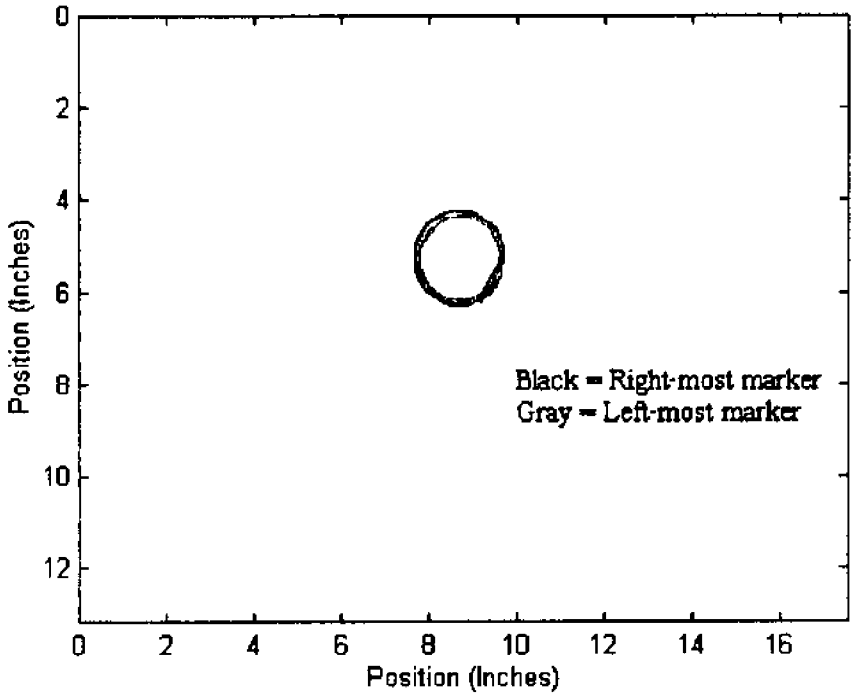


Figure 6: Position of the two markers
Test: Spinning Disk

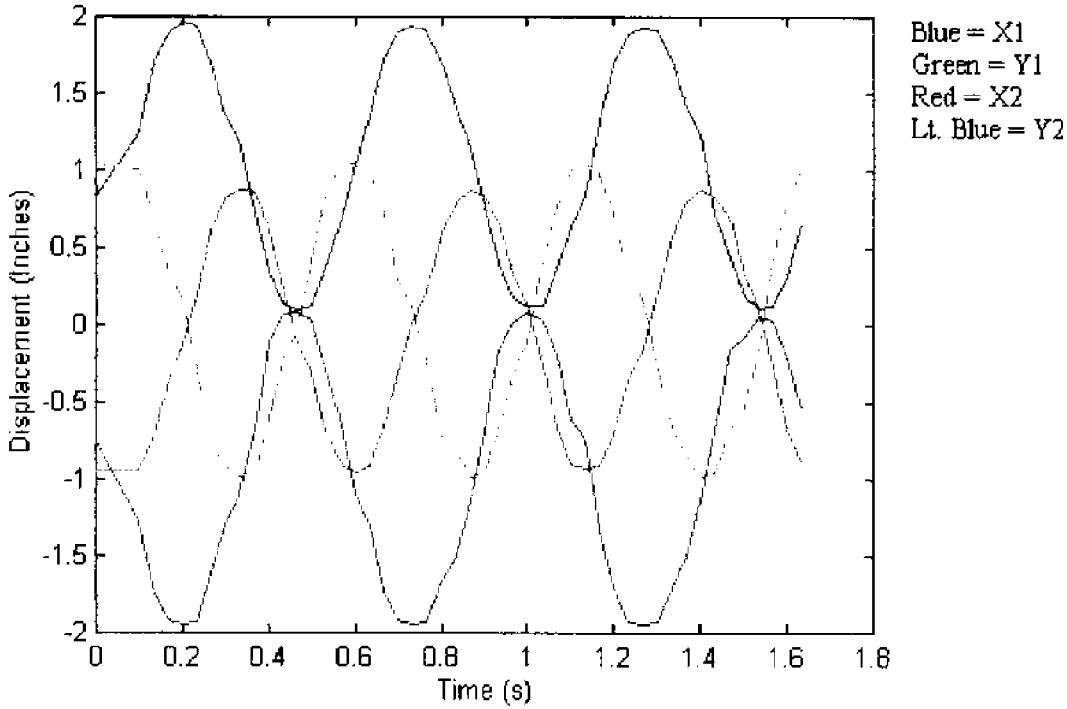


Figure 7: Vertical and Horizontal Displacement from Equilibrium
Test: Spinning Disk

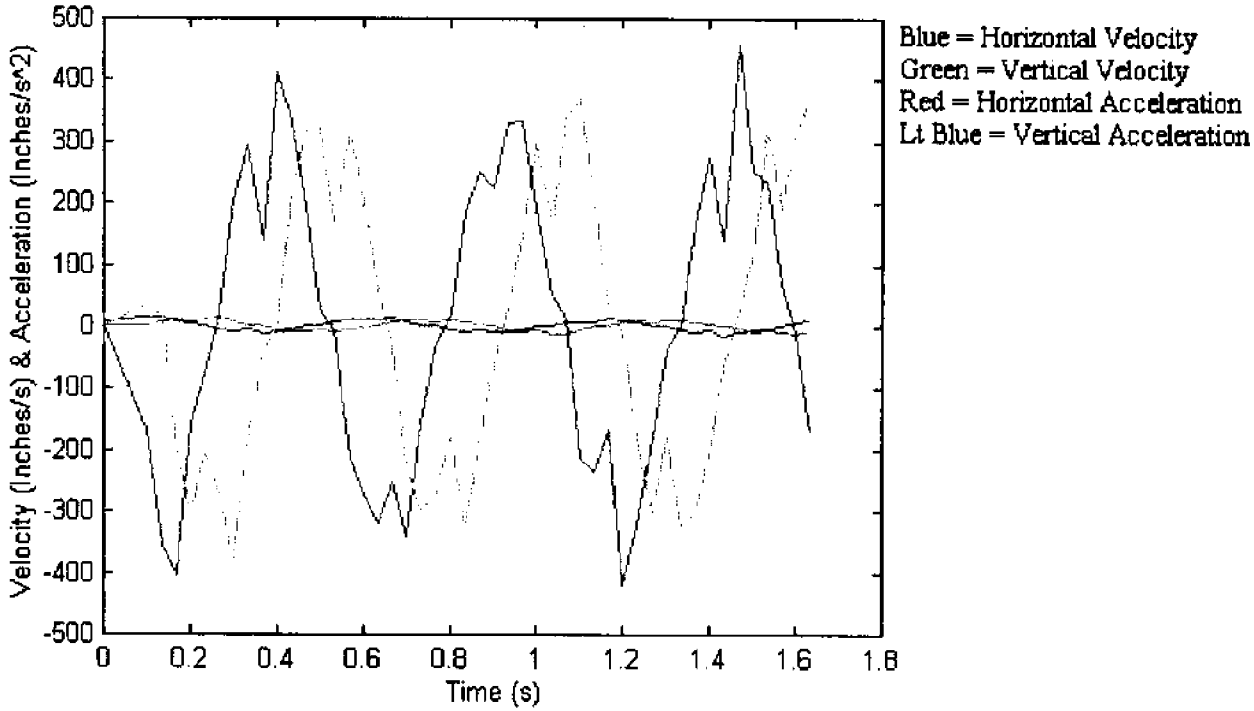


Figure 8: Velocity and Acceleration Components of the First Point Test: Spinning Disk

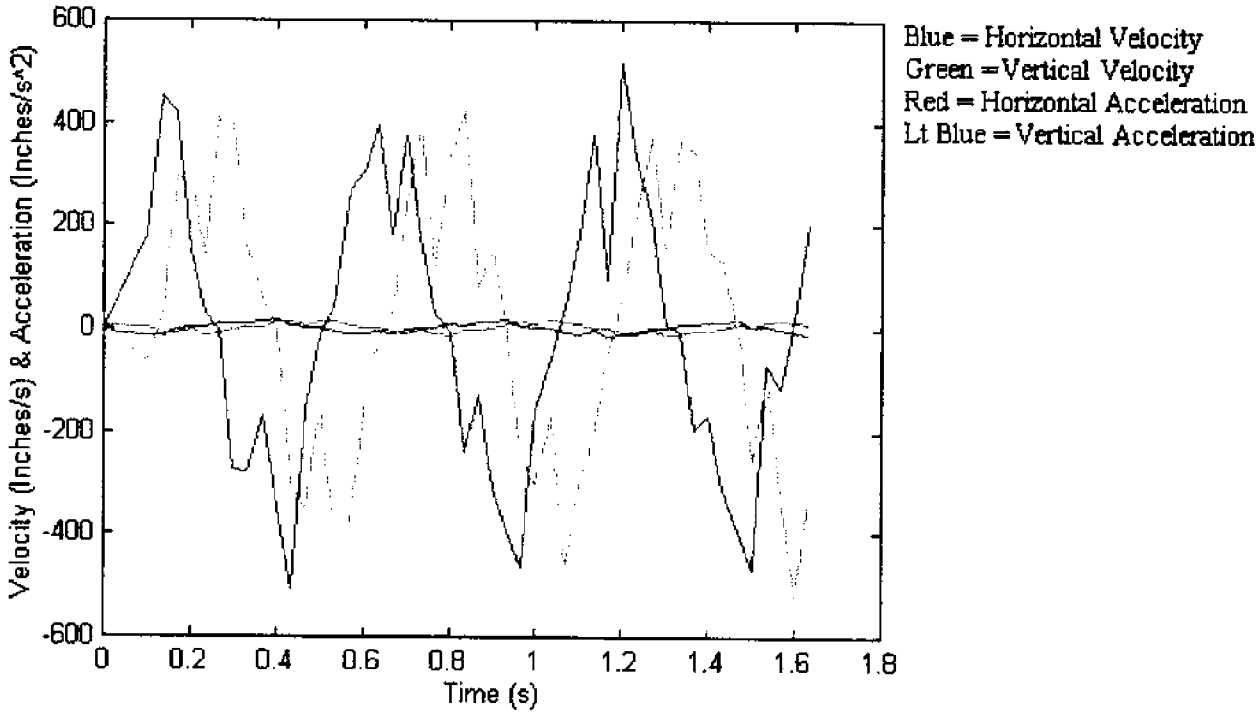


Figure 9: Horizontal and Vertical Velocities and Accelerations for the Second Point

Test: Spinning Disk

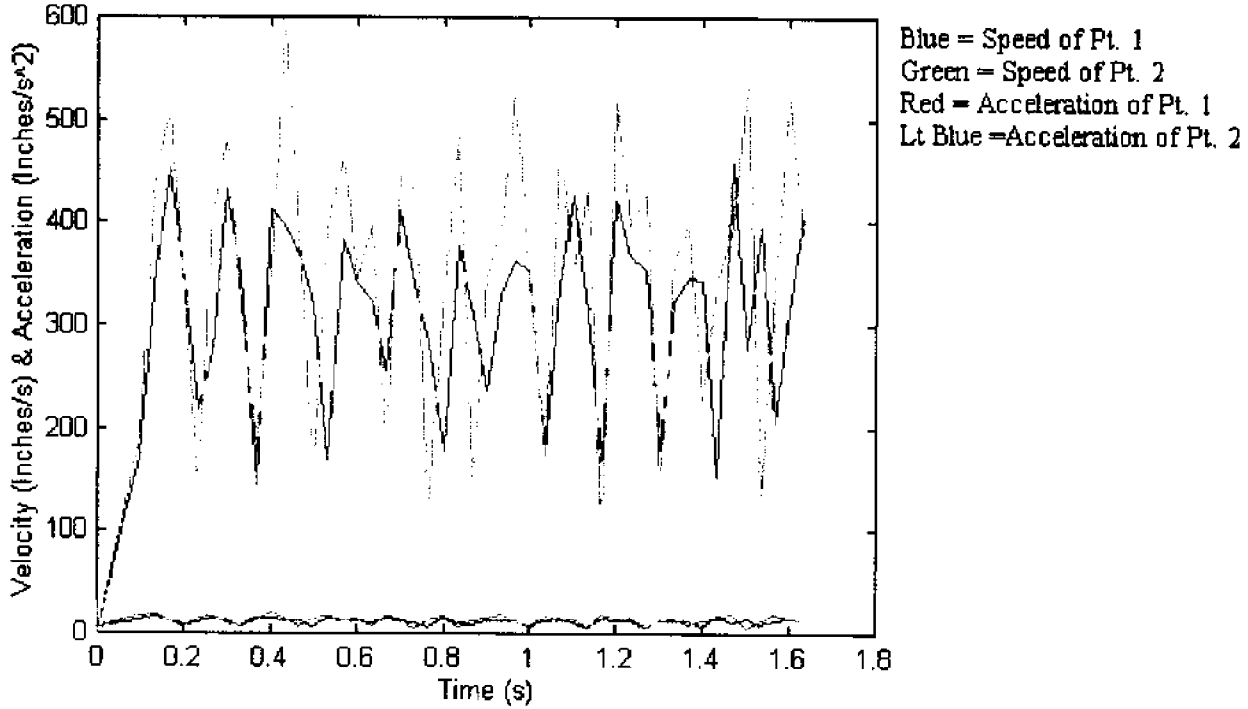


Figure 10: Speed and Total Acceleration of Both Points
Test: Spinning Disk

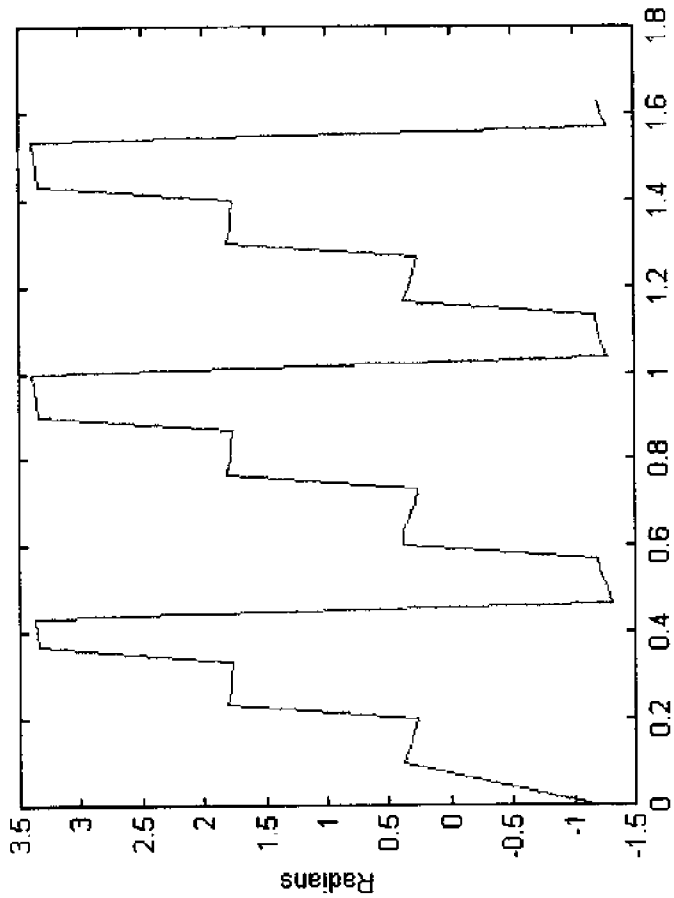


Figure 11: Angular Displacement of the Disk Test. Spinning Disk

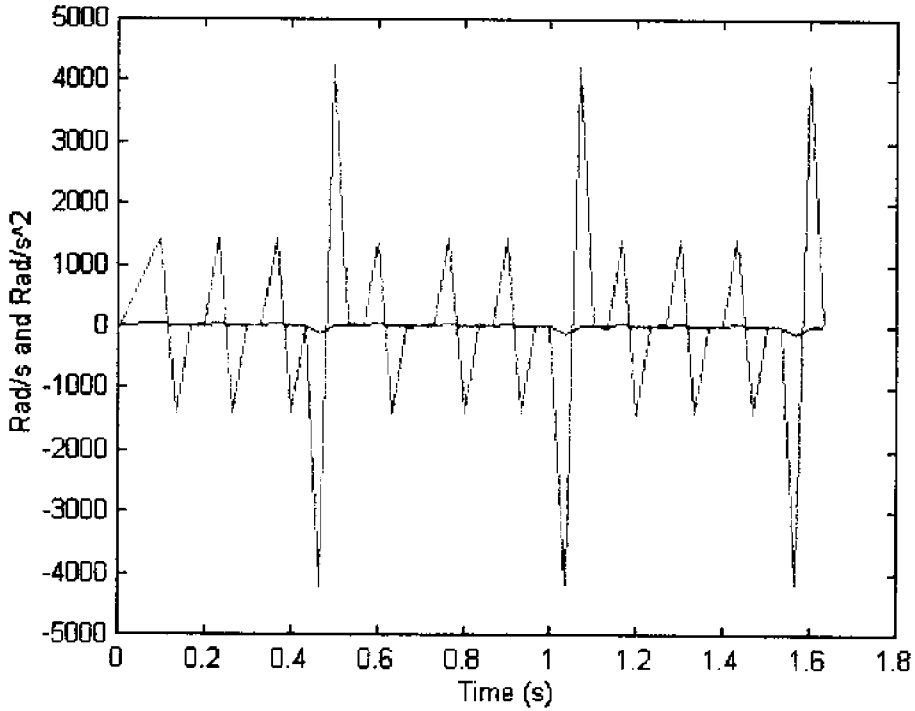


Figure 12: Angular Velocity and Acceleration of the Disk

Test: Spinning Disk

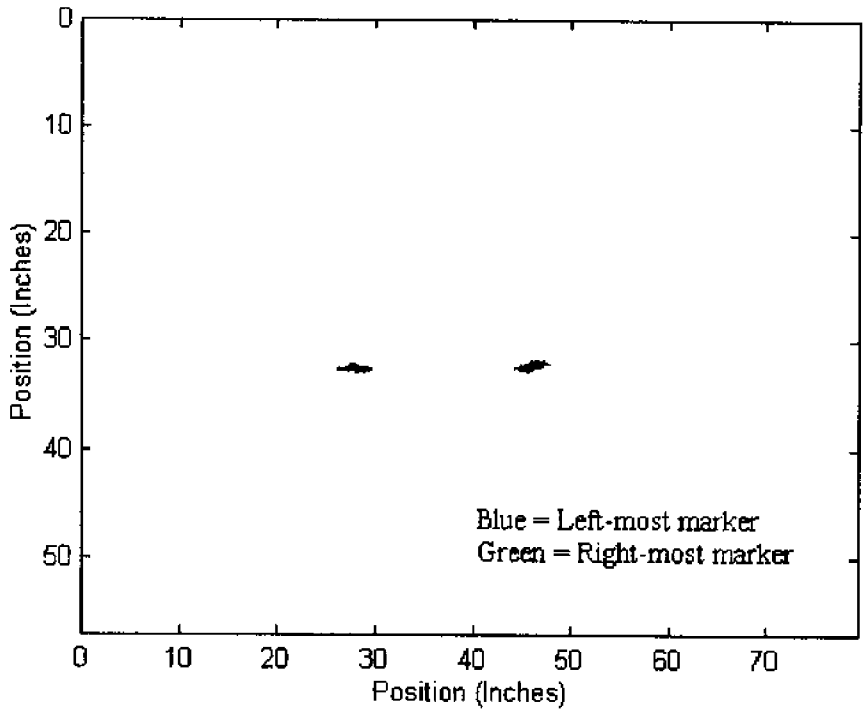


Figure 13: Position of the Markers
Test: Fish Cage

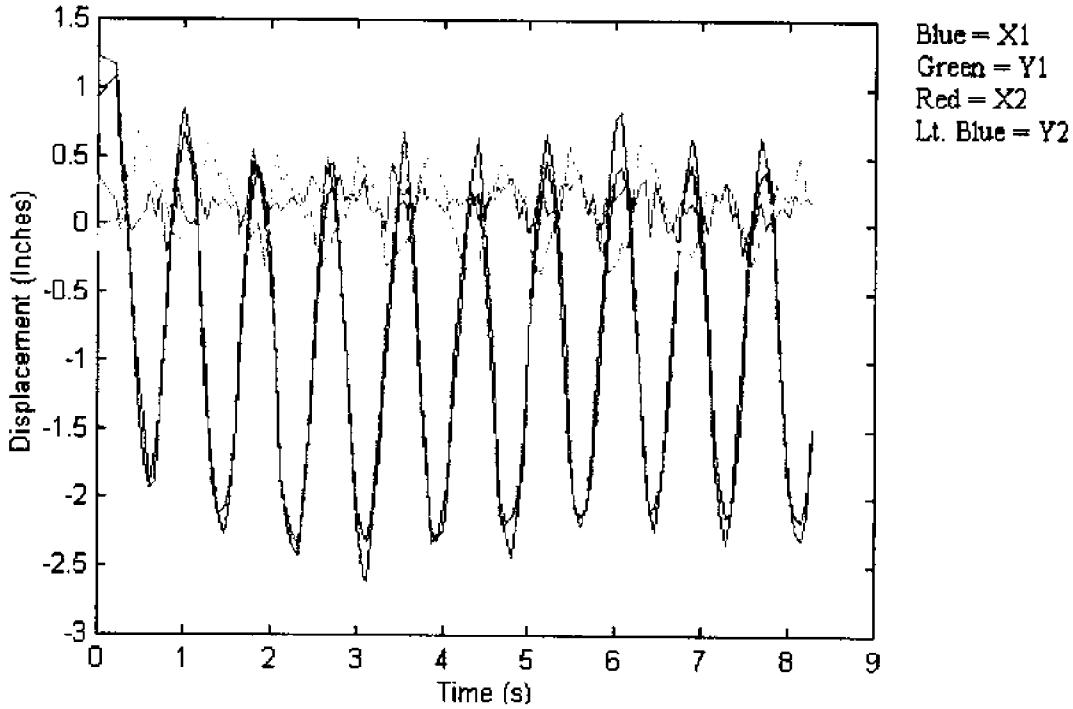


Figure14: Horizontal and Vertical Displacement from Equilibrium
Test: Fish Cage

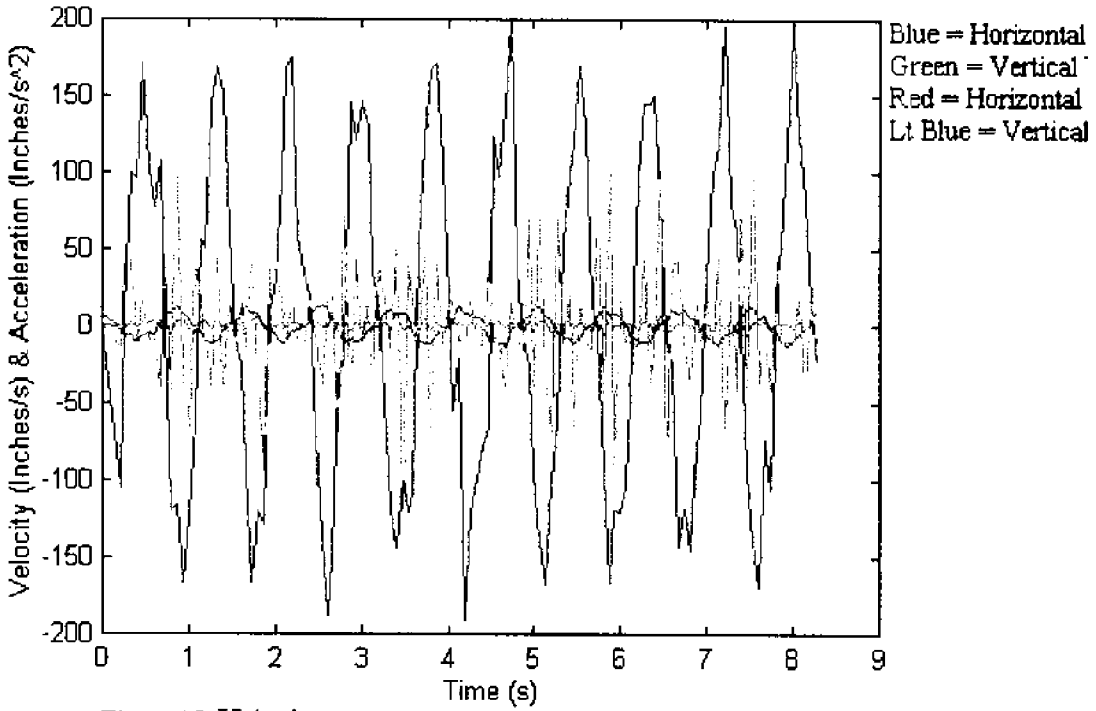


Figure 15: Velocity and Acceleration Components of the First Point Test: Fish Cage

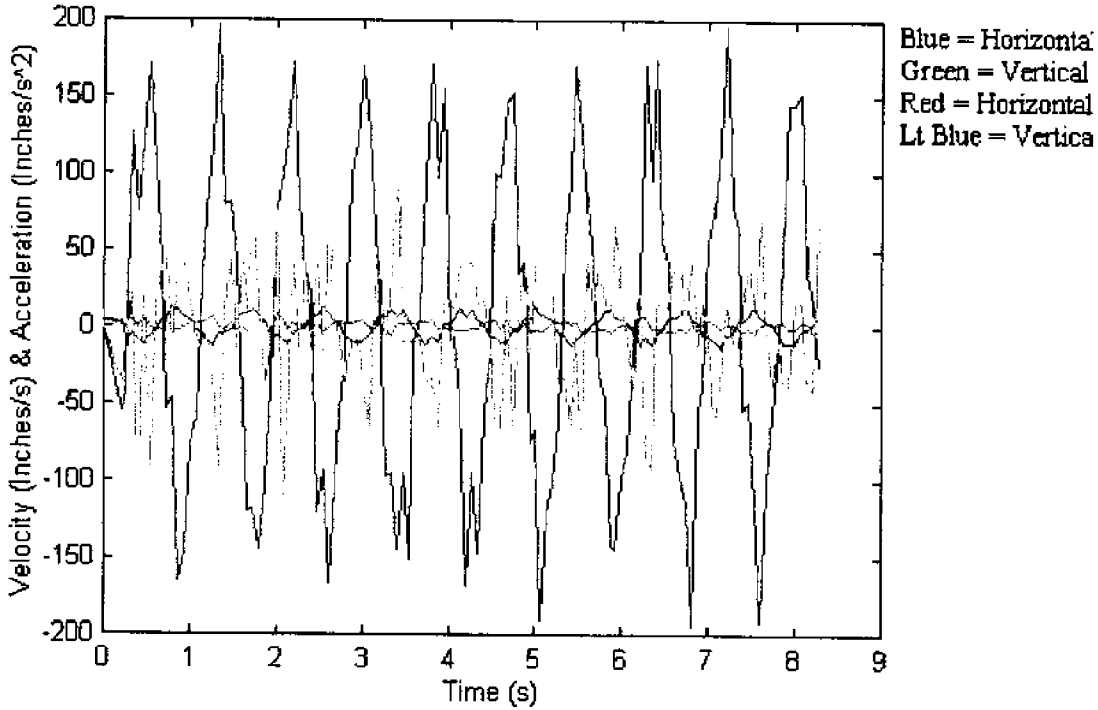


Figure 16: Horizontal and Vertical Velocities and Accelerations for the Second Point
Test: Fish Cage

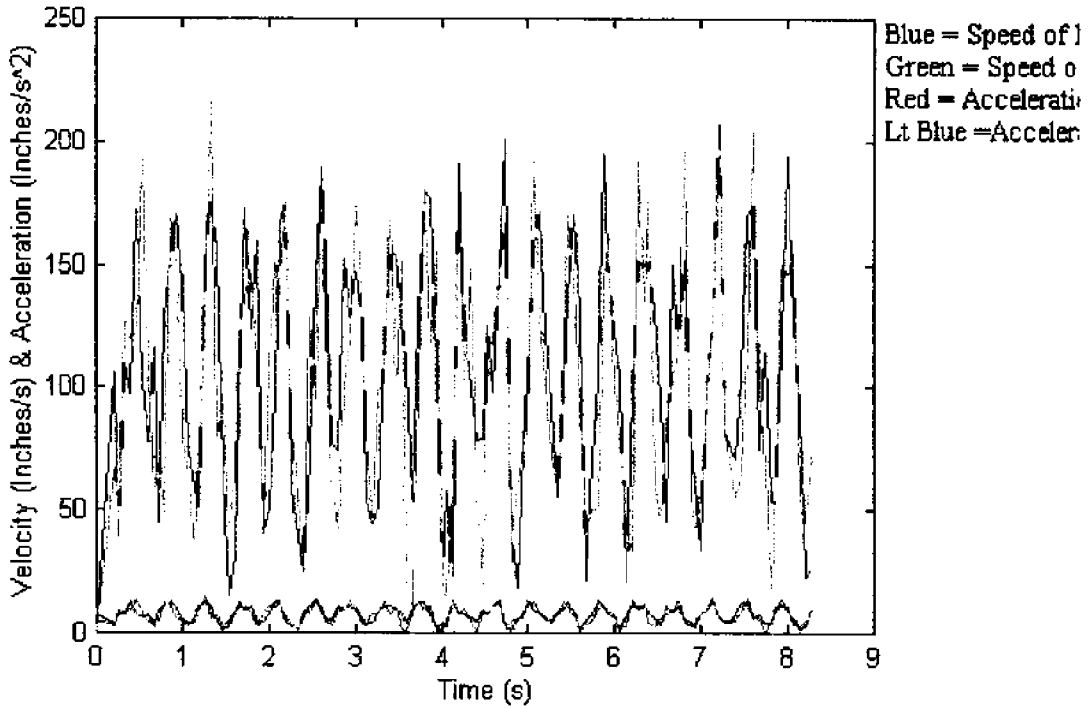


Figure 17: Speed and Total Acceleration of Both Points
Test: Fish Cage

Appendix C: OPIE CODE

```
*****************************************************************
*    OPIE Video Analysis Software
*    Created by:  Derek Michelin and Shannon Stott
*    Funded through:  SeaGrant and UNH COE
*****************************************************************
%This function begins the user interface window

%Assigns a handle to the window and clears it

clc

%Text that identifies program
txt = uicontrol(gcf,'Style','text','Position',[190 330 120 35],...
'String','O.P.I.E. Analysis Program','BackgroundColor',[0 0 0],...
'ForegroundColor',[1 0 0]);

%Cover image is loaded and displayed in the current figure
[A,map]=bmpread('C:\matlab\bin\video\cal0001.bmp');
imshow(A,map);

%Creates default 'quit' button
opt1=uicontrol(gcf,'Style','pushbutton','String',...
'Quit','Position',[75 10 120 25],'Callback','close(gcf)');

%Creates button to start the OPIE program/Activates Program
anl_opt=uicontrol(gcf,'Style','pushbutton','String',...
'Start Analysis','Position',[300 10 120 25],'Callback','close(gcf),info');
```

```
*****************************************************************
*       First CallBack of the OPIE Program
*       User Defines all of the Test Parameters
*****************************************************************
%Clears all variables in Matlab's memory
clear;
clc;
k=1;
count=1;

backcolor = uicontrol(gcf,'Style','frame','Pos',[0 0 1
1],'units','normalized',...
'BackgroundColor',[1 1 1]);

txt = uicontrol(gcf,'Style','text','Position',[145 320 180 20],...
'String','Testing Parameters','BackgroundColor',[1 1 1],...
'ForegroundColor',[0 0 0]);

input_frame = uicontrol(gcf,'Style','frame','Pos',[95 70 295 215],...
'BackgroundColor',[.701961 .701961 .701961]);

fname_txt=uicontrol(gcf,'Style','text','pos',[105 250 215 15],'String',...
'Enter Sequence File Name','BackgroundColor',[.701961 .701961
.701961],'Horiz','Left');

shade = uicontrol(gcf,'Style','frame','pos',[322 247 54 21],...
'BackgroundColor',[1 1 1]);

FNAME_val =uicontrol(gcf,'Style','edit','pos',[325 250 50 15],...
'BackgroundColor',[1 1 1],'CallBack',['get(FNAME_val,''String'');',...
'fname = get(FNAME_val,''String'');']);

cname_txt=uicontrol(gcf,'Style','text','pos',[105 175 215 15],'String',...
'Enter Name of Calibration Image','BackgroundColor',[.701961 .701961
.701961],'Horiz','Left');

shade2 = uicontrol(gcf,'Style','frame','pos',[322 172 54 21],...
'BackgroundColor',[1 1 1]);

cname_val =uicontrol(gcf,'Style','edit','pos',[325 175 50 15],...
'BackgroundColor',[1 1 1],'CallBack',['get(cname_val,''String'');',...
'cname = get(cname_val,''String'');']);
```

```
diam_txt=uicontrol(gcf,'Style','text','pos',[105 100 215 15],'String',...
'Enter diameter of calibration circle','BackgroundColor',...
[.701961 .701961 .701961],'Horiz','Left');

shade3 = uicontrol(gcf,'Style','frame','pos',[322 97 54 21],...
'BackgroundColor',[1 1 1]);

diam_val=uicontrol(gcf,'Style','edit','pos',[325 100 50 15],...
'BackgroundColor',[1 1 1],'CallBack',['get(diam_val,''String'');',...
'diam = str2num(get(diam_val,''String''));']);

ename_txt=uicontrol(gcf,'Style','text','pos',[105 125 215 15],'String',...
'Enter Name of Equilibrium Image','BackgroundColor',[.701961 .701961
.701961],'Horiz','Left');

shade4 = uicontrol(gcf,'Style','frame','pos',[322 122 54 21],...
'BackgroundColor',[1 1 1]);

ename_val =uicontrol(gcf,'Style','edit','pos',[325 125 50 15],...
'BackgroundColor',[1 1 1],'CallBack',['get(ename_val,''String'');',...
'ename = get(ename_val,''String'');']);

f_txt=uicontrol(gcf,'Style','text','pos',[105 225 215 15],'String',...
'Number of Frames in Sequence','BackgroundColor',[.701961 .701961
.701961],...
'Horiz','Left');

shade5 = uicontrol(gcf,'Style','frame','pos',[322 222 54 21],...
'BackgroundColor',[1 1 1]);

F_val=uicontrol(gcf,'Style','edit','pos',[325 225 50 15],...
'BackgroundColor',[1 1 1],'CallBack',['get(F_val,''String'');',...
'f=str2num(get(F_val,''String''));']);

numpnt_txt=uicontrol(gcf,'Style','text','pos',[105 150 215 15],'String',...
'Enter Number of Markers on Model','BackgroundColor',...
[.701961 .701961 .701961],'Horiz','Left');

shade6 = uicontrol(gcf,'Style','frame','pos',[322 147 54 21],...
'BackgroundColor',[1 1 1]);

numpnt_val=uicontrol(gcf,'Style','edit','pos',[325 150 50 15],...
'BackgroundColor',[1 1 1], 'Value', 2,'CallBack',
['get(numpnt_val,''String'');',...
'numpnt=str2num(get(numpnt_val,''String''));']);
```

```
fps_txt = uicontrol(gcf,'Style','text','pos',[105 200 160 15],'String',...
'Sampling Rate (frames/sec)','BackgroundColor',[.701961 .701961
.701961]);

shade = uicontrol(gcf,'Style','frame','pos',[322 197 54 21],...
'BackgroundColor',[1 1 1]);

FPS_val = uicontrol(gcf,'Style','edit','pos',[325 200 50 15],...
'BackgroundColor',[1 1 1],'Value',30,'CallBack',['get(FPS_val,''String'');',...
'fps = str2num(get(FPS_val,''String''));']);

quit = uicontrol(gcf,'Style','pushbutton','String',...
'Quit','Position',[15 5 115 25],'BackgroundColor',[.9 1 1],'Callback',...
['close(gcf)',',Return,']);

%Activates calibration (next step)
finished = uicontrol(gcf,'Style','pushbutton','String',...
'Continue','Position',[370 5 115 25],'BackgroundColor',[.9 1
1],'Callback',...
['close(gcf),calb']);
```

```
*****************************************************************
*   Second CallBack of OPIE:  Calibrates the pixel info
*****************************************************************

% Text that identifies program
txt = uicontrol(gcf,'Style','text','Position',[15 340 120 20],...
'String','Calibration','BackgroundColor',[0 0 0],...
'ForegroundColor',[1 0 0]);

cfile=['C:\matlab\bin\video\',cname,'0001.bmp'];
[A,map]=bmpread(cfile);
imshow(A,map);

Inten_txt=uicontrol(gcf,'Style','text','pos',[165 35 100 15],'String',...
'Intensity','BackgroundColor',[.701961 .701961 .701961]);

Inten_val=uicontrol(gcf,'Style','slider','pos',[185 15 120 20],'min',1,...
'max',256,'Value',150,'CallBack',['set(Inten,''String'',',...
'num2str(get(Inten_val,''Val'')))',']);

Inten=uicontrol(gcf,'Style','text','pos',[265 35 62 15],...
'BackgroundColor',[1 1 1],'String',num2str(get(Inten_val,'Value')));

Inten_min=uicontrol(gcf,'Style','text','pos',[165 15 20
20],'BackgroundColor',...
[.701961 .701961 .701961],'String',...
num2str(get(Inten_val,'min')));

Inten_max=uicontrol(gcf,'Style','text','pos',[305 15 22 20],...
'BackgroundColor',[.701961 .701961 .701961],'String',...
num2str(get(Inten_val,'max')),'Horiz','Right');

frame_txt = uicontrol(gcf,'Style','text','Position',[375 350 130 25],...
'String','Currently Viewing Frame: ','BackgroundColor',[0 0 0],...
'ForegroundColor',[1 .8 1],'HorizontalAlignment','Left');

frame_txt2 = uicontrol(gcf,'Style','text','Position',[420 330 25 20],...
'String',cname,'BackgroundColor',[0 0 0],...
'ForegroundColor',[1 1 .8]);

refresh=uicontrol(gcf,'Style','pushbutton','String',...
'Run Filter','Position',[370 35 115 25],'Callback',...
['but']);
```

```
cal_circle = uicontrol(gcf,'Style','pushbutton','String',...
'Select Circle','Position',[15 35 115 25],'Callback',...
['but']);

quit = uicontrol(gcf,'Style','pushbutton','String',...
'Quit','Position',[15 5 115 25],'BackgroundColor',[.9 1 1],'Callback',...
['close(gcf)',',Return,']);

%Activates equilibrium
finished = uicontrol(gcf,'Style','pushbutton','String',...
'Finished','Position',[370 5 115 25],'BackgroundColor',[.9 1 1],'Callback',...
['close(gcf),equi,imzoom off']);
```

```
**************************************************************
* Third Callback:  Reference Point Determination
**************************************************************
% Text that identifies program
txt = uicontrol(gcf,'Style','text','Position',[25 340 225 20],...
'String','Selection of the Two Equilibrium Points ','BackgroundColor',
[0 0 0],'ForegroundColor',[1 1 1]);

frame_txt = uicontrol(gcf,'Style','text','Position',[375 350 130 25],...
'String','Currently Selecting From Frame:  ','BackgroundColor',...
[0 0 0],'ForegroundColor',[1 .8 1],'HorizontalAlignment','Left');

frame_txt2 = uicontrol(gcf,'Style','text','Position',[420 330 25 20],...
'String',ename,'BackgroundColor',[0 0 0],...
'ForegroundColor',[1 1 .8]);

efile = ['C:\matlab\bin\video\',ename,'0001.bmp'];

[A,map] = bmpread(efile);
imshow(A,map);

refresh = uicontrol(gcf,'Style','pushbutton','String',...
'Clear Points','Position',[15 35 115 25],'Callback',...
['',imshow(A,map),']);

quit = uicontrol(gcf,'Style','pushbutton','String',...
'Quit','Position',[15 5 115 25],'BackgroundColor',[.9 1 ],'Callback',...
['close(gcf)',',Return,']);

zm_on = uicontrol(gcf,'Style','checkbox','String','Zoom On',...
'Pos',[140 35 90 25],'Value',0,'BackgroundColor',[.701961 .701961
.701961],...
'CallBack',['set(zm_on,''Value'',1),','set(zm_off,''Value'',0),','imzoom on']);

zm_off = uicontrol(gcf,'Style','checkbox','String','Zoom Off',...
'Pos',[140 5 90 25],'Value',1,'BackgroundColor',[.701961 .701961
.701961],...
'CallBack',['set(zm_on,''Value'',0),',...
'set(zm_off,''Value'',1),','imzoom off']);

lft_pnt = uicontrol(gcf,'Style','pushbutton','String',...
'Select first point','Position',[240 35 120 25],'BackgroundColor',[.9 1 1],...
'Callback',['[B(k,1),B(k,2)] = ginput(1);',',imzoom
out,',',xref(1) = B(k,1);,',',xref(2) = B(k,2);,']);
```

```
rt_pnt = uicontrol(gcf,'Style','pushbutton','String',...
'Select second point','Position',[240 5 120 25],'BackgroundColor',[.9 1
1],'Callback',['[B(k,3),B(k,4)] = ginput(1);',',imzoom
out,',',xref(3) = B(k,3);,',',xref(4) = B(k,4);,']);

show_pnts = uicontrol(gcf,'Style','pushbutton','String',...
'Show points','Position',[370 35 115 25],'Callback',...
[',point,']);

%Activates Point Selection
finished = uicontrol(gcf,'Style','pushbutton','String',...
'Finished','Position',[370 5 115 25],'BackgroundColor',[.9 1 1],'Callback',...
['close(gcf),pick']);
```

```
*************************************************************
*  This callback allows the user to easily select the first two
*  position markers on the OUT
*************************************************************

txt = uicontrol(gcf,'Style','text','Position',[25 340 225 20],...
'String','Selection of the two position markers','BackgroundColor',[0 0 0],...
'ForegroundColor',[1 1 1]);

frame_txt = uicontrol(gcf,'Style','text','Position',[375 350 130 25],...
'String','Currently Selecting From Frame Number:  ','BackgroundColor',...
[0 0 0],'ForegroundColor',[1 .8 1],'HorizontalAlignment','Left');

frame_txt2 = uicontrol(gcf,'Style','text','Position',[420 330 25 20],...
'String',int2str(k),'BackgroundColor',[0 0 0],...
'ForegroundColor',[1 1 .8]);

file = ['C:\matlab\bin\video\',fname,'000',int2str(k),'.bmp'];
[A,map] = bmpread(file);
imshow(A,map);

refresh = uicontrol(gcf,'Style','pushbutton','String',...
'Clear Points','Position',[15 35 115 25],'Callback',...
[',imshow(A,map),']);

quit = uicontrol(gcf,'Style','pushbutton','String',...
'Quit','Position',[15 5 115 25],'BackgroundColor',[.9 1 1],'Callback',...
['close(gcf)',',Return,']);

zm_on = uicontrol(gcf,'Style','checkbox','String','Zoom On',...
'Pos',[140 35 90 25],'Value',0,'BackgroundColor',[.701961 .701961
.701961],...
'CallBack',['set(zm_on,''Value'',1),',',set(zm_off,''Value'',0),',',imzoom on']);

zm_off = uicontrol(gcf,'Style','checkbox','String','Zoom Off',...
'Pos',[140 5 90 25],'Value',1,'BackgroundColor',[.701961 .701961
.701961],...
'CallBack',['set(zm_on,''Value'',0),',...
'set(zm_off,''Value'',1),',',imzoom off']);

lft_pnt = uicontrol(gcf,'Style','pushbutton','String',...
'Select first point','Position',[240 35 120 25],'BackgroundColor',[.9 1 1],...
'Callback',['[B(k,1),B(k,2)] = ginput(1);',',imzoom out,']);
```

```
rt_pnt = uicontrol(gcf,'Style','pushbutton','String',...
'Select second point','Position',[240 5 120 25],'BackgroundColor',[.9 1 1],.
'Callback',['[B(k,3),B(k,4)] = ginput(1);',',imzoom out,']);

show_pnts = uicontrol(gcf,'Style','pushbutton','String',...
'Show points','Position',[370 35 115 25],'Callback',...
[',point,']);

%Starts final analysis
finished = uicontrol(gcf,'Style','pushbutton','String',...
'Finished','Position',[370 5 115 25],'BackgroundColor',[.9 1 1],'Callback',...
['close(gcf)',',analysis,']);

*****************************************************************
*Callback within callback : points
*Draws cross hairs on selected point
*****************************************************************
[m,n] = size(A);
numpnt = 2;
%Shows location of point
      count = 1;
      C = A;
      while count < 2*numpnt,
            j = 1;
            i = 1;
            while j > = 1 & j < = n,
                  C(B(k,count + 1),j) = 1;
                  C(B(k,count + 1) + 1,j) = 1;
                  j = j + 1;
                  end
            while i > = 1 & i < = m,
                  C(i,B(k,count)) = 1;
                  C(i,B(k,count) + 1) = 1;
                  i = i + 1;
                  end
            C(B(k,count + 1),B(k,count)) = 256;
            count = count + 2;
            end
      imshow(C,map);
      drawnow;
```

```
*****************************************************************
*       Final analysis/position, velocity and acceleration
*****************************************************************
k=2;
t(1)=0;
while k<=f,
        if k<10
                file=['C:\matlab\bin\video\',fname,'000',int2str(k),'.bmp'];
        elseif k>9 & k<100
                file=['C:\matlab\bin\video\',fname,'00',int2str(k),'.bmp'];
        elseif k>99 & k<1000
                file=['C:\matlab\bin\video\',fname,'0',int2str(k),'.bmp'];
        elseif k>999 & k<10000
                file=['C:\matlab\bin\video\',fname,int2str(k),'.bmp'];
        else
                Error='Do you know how long that would take?'
                return
                end
        [A,map]=bmpread(file);
        [m,n]=size(A);
        n=n-11;

        d=sqrt((B(1,3)-B(1,1))^2+(B(1,4)-B(1,2))^2);
        dact=sqrt(((B(1,3)-B(1,1))*calx)^2+((B(1,4)-B(1,2))*caly)^2);


%Finds predicted values
                count=1;
                while count<=2*numpnt,
                        b(count)=B(k-1,count);
                        count=count+1;
                        end
```

```
%Finds points near predicted value
            count = 1;
            while count < = 2*numpnt-1,
                  j = b(count)-d/2;
                  min = 256;
                  while j< = b(count) + d/2 & j< = n & j> = 1,
                        i = b(count + 1)-d/2;
                        while i< = b(count + 1) + d/2 & i< = m & i> = 1,
                              rad = sqrt((b(count)-j)^2 + (b(count + 1)-i)^2);
                              if rad < = d/2 & A(i,j) < min
                                    B(k,count) = j;
                                    B(k,count + 1) = i;
                                    min = A(i,j);
                                    end
                              i = i + 1;
                              end
                        j = j + 1;
                        end
                  count = count + 2;
                  end


%Shows location of point
      count = 1;
      C = A;
      while count < 2*numpnt,
            j = 1;
            i = 1;
            while j> = 1 & j< = n,
                  C(B(k,count + 1),j) = 1;
                  C(B(k,count + 1) + 1,j) = 1;
                  j = j + 1;
                  end

            while i> = 1 & i< = m,
                  C(i,B(k,count)) = 1;
                  C(i,B(k,count) + 1) = 1;
                  i = i + 1;
                  end
            C(B(k,count + 1),B(k,count)) = 256;
            count = count + 2;
            end

txt = uicontrol(gcf,'Style','text','Position',[25 340 225 20],...
'String','O.P.I.E. Position Locator','BackgroundColor',[0 0 0],...
'ForegroundColor',[1 1 1]);
```

```
frame_txt = uicontrol(gcf,'Style','text','Position',[375 350 130 25],...
'String','Currently Viewing Frame: ','BackgroundColor',[0 0 0],...
'ForegroundColor',[1 .8 1],'HorizontalAlignment','Left');

frame_txt2 = uicontrol(gcf,'Style','text','Position',[420 330 25 20],...
'String',int2str(k),'BackgroundColor',[0 0 0],...
'ForegroundColor',[1 1 .8]);

quit=uicontrol(gcf,'Style','pushbutton','String',...
'Quit','Position',[200 5 115 25],'BackgroundColor',[.9 1 1],'Callback',...
['close(gcf)',',Return,']);


        imshow(C,map);
        drawnow;
        k=k+1;
        t(k+1)=k/fps;
        end

for k=1:f
        for count=1:4
                B(k,count)=B(k,count)*cal(count);
                end
        end

for k=1:f
        for count=1:4
                if k==1
                        D(1,count)=B(2,count)-xref(count)*cal(count);;
                else
                        D(k,count)=B(k,count)-xref(count)*cal(count);
                        end
                end
        end

for count=1:4
        vel(1,count)=(B(2,count)-B(1,count))*fps;;
        end
```

```
for k = 1:3
        for count = 1:4
                accel(k,count) = 0;
                end
        end
for k = 2:1:f
        for count = 1:1:4
                vel(k,count) = (B(k,count)-B(k-1,count)) * fps;
                if k > = 4
                        accel(k,count) = vel(k,count)-vel(k-1,count) * fps;
                        end
                end
        end

for k = 1:f
        veltot1(k) = sqrt(vel(k,1) ^ 2 + vel(k,2) ^ 2);
        veltot2(k) = sqrt(vel(k,3) ^ 2 + vel(k,4) ^ 2);
        acctot1(k) = sqrt(accel(k,1) ^ 2 + accel(k,2) ^ 2);
        acctot2(k) = sqrt(accel(k,3) ^ 2 + accel(k,4) ^ 2);
        end

if xref(1) < xref(3) & xref(2) < = xref(4)
        theta_ref = asin((xref(4)-xref(2))/dact);
        end
if xref(1) > = xref(3) & xref(2) < xref(4)
        theta_ref = 3.14159/2 + asin((xref(4)-xref(2))/dact);
        end
if xref(1) > xref(3) & xref(2) > = xref(4)
        theta_ref = 3.14159 + asin((xref(4)-xref(2))/dact);
        end
if xref(1) < = xref(3) & xref(2) > xref(4)
        theta_ref = 3 * 3.14159/2 + asin((xref(4)-xref(2))/dact);
        end

for k = 1:f
        if B(k,1) < B(k,3) & B(k,2) < = B(k,4)
                theta(k) = asin(((B(k,4)-B(k,2))/dact)-theta_ref);
                end
        if B(k,1) > = B(k,3) & B(k,2) < B(k,4)
                theta(k) = 3.14159/2 + asin(((B(k,4)-B(k,2))/dact)-theta_ref);
                end
        if B(k,1) > B(k,3) & B(k,2) > = B(k,4)
                theta(k) = 3.14159 + asin(((B(k,4)-B(k,2))/dact)-theta_ref);
                end
        if B(k,1) < = B(k,3) & B(k,2) > B(k,4)
                theta(k) = 3 * 3.14159/2 + asin(((B(k,4)-B(k,2))/dact)-theta_ref);
```

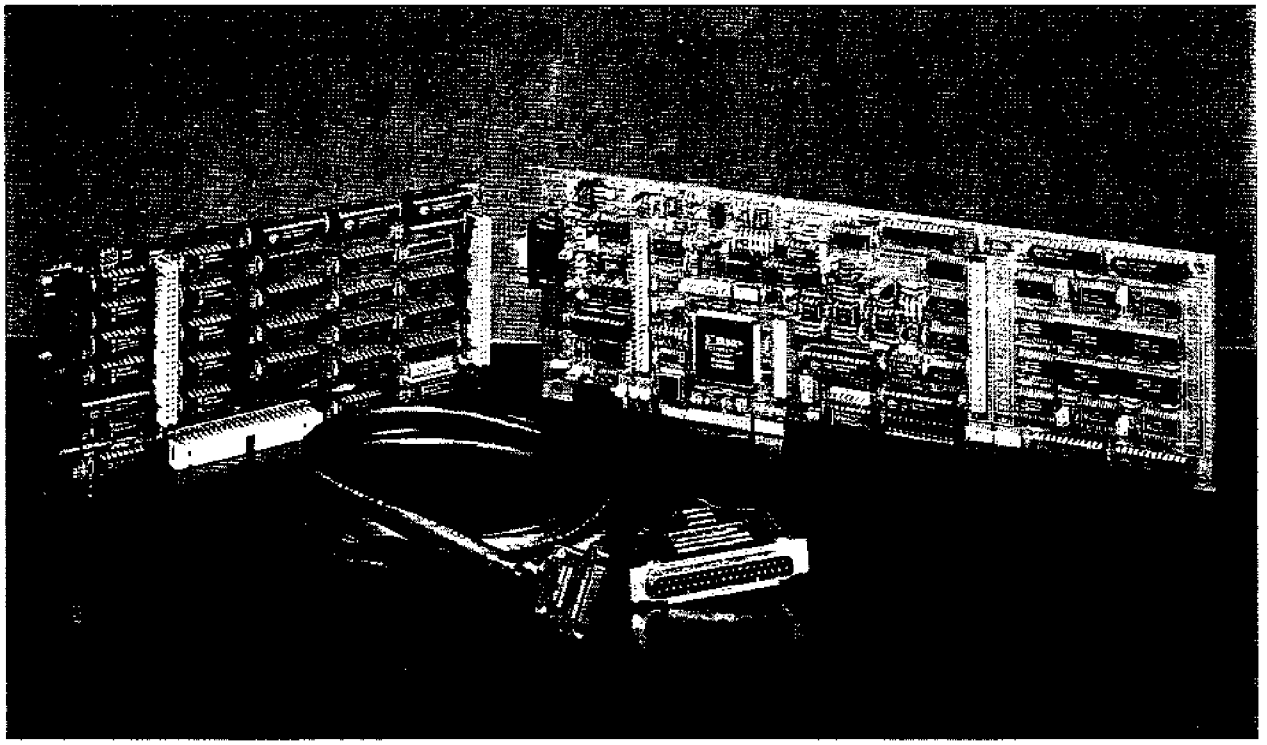
```
            end

        end

omega(1) = (theta(2)-theta(1))*fps;
alpha(1) = 0;
for k = 2:f
        omega(k) = (theta(k)-theta(k-1))*fps;
        alpha(k) = (omega(k)-omega(k-1))*fps;
        end

units = ['inches'];

figure
plot(B(1:f,1),B(1:f,2),'-',B(1:f,3),B(1:f,4),'-');
axis([0,n*cal(1),0,m*cal(2)]);
axis('ij');
title('Position in Frame');
xaxis = ['Position (',units,')'];
xlabel(xaxis);
yaxis = ['Position (',units,')'];
ylabel(yaxis);


figure
plot(t(1:f),D(1:f,1),t(1:f),D(1:f,2),t(1:f),D(1:f,3),t(1:f),D(1:f,4));
axis('xy');
title('Displacement from Equilibrium');
xlabel('Time (s)');
yaxis = ['Displacement (',units,')'];
ylabel(yaxis);


figure
plot(t(1:f),vel(1:f,1),t(1:f),vel(1:f,2),t(1:f),accel(1:f,1),t(1:f),accel(1:f,2));
title('Velocty and Acceleration Components of the First Point');
xlabel('Time (s)');
yaxis = ['Velocity (',units,'/s) & Acceleration (',units,'/s^2)'];
ylabel(yaxis);

figure
plot(t(1:f),vel(1:f,3),t(1:f),vel(1:f,4),t(1:f),accel(1:f,3),t(1:f),accel(1:f,4));
title('Velocity and Acceleration Components of the Second Point');
xlabel('Time (s)');
yaxis = ['Velocity (',units,'/s) & Acceleration (',units,'/s^2)'];
ylabel(yaxis);
```

```
figure
plot(t(1:f),veltot1(1:f),t(1:f),veltot2(1:f),t(1:f),acctot1(1:f),t(1:f),acctot2(1:f))
;
title('Speed and Acceleration for Both Points')
xlabel('Time (s)');
yaxis=['Velocity (',units,'/s) & Acceleration (',units,'/s^2)'];
ylabel(yaxis);

figure
plot(t(1:f),theta(1:f));
title('Angular Displacement')
xlabel('Time (s)');
ylabel('Radians');

figure
plot(t(1:f),omega(1:f),t(1:f),alpha(1:f));
title('Angular Velocity and Acceleration')
xlabel('Time (s)');
ylabel('Rad/s and Rad/s^2');
```

Appendix D: Product Specification Sheets

M-Vision 1000

PCI Local Bus Video Digitizer



Applications

- Machine/Industrial Vision
- Image Analysis
- Medical Imaging
- Motion Analysis

Features

- Single slot PCI Local Bus Frame Grabber
- 0-40 M samples/second digitization
- Digital Input Module (optional MV- 1100)
- External I/O Controls
- Configurable Memory
- Plug and Play Auto Configuration
- Slave and Master Mode Data Transfer

Software

- All boards come with DOS and Windows utilities that allow the user to grab and save images in .TIF, .TGA, .BMP, file formats
- The M-Vision 1000 product line includes DOS, Windows 3.1 and Windows NT development libraries
- Many third party software packages offer M-Vision 1000 compatible drivers — Contact MuTech

The M-VISION 1000 (MV-1000) is a single slot video digitizer board for the PCI (Peripheral Component Interconnect) bus. The MV-1000 digitizes standard or non-standard analog camera video into 8 bits per pixel at rates up to 40 million samples per second. An optional 10 bit version is available for ultra high quality video applications. The digitized video is stored in on-board VRAM or transferred in real time to system memory and/or the VGA card for display. Up to four cameras may be attached to a single MV-1000.

The optional Digital Camera Interface (MV-1100) supports single ended and differential input from 8, 16, and 32 bit devices. The MV-1100 is an excellent solution for interfacing to the new generation of high speed digital RGB cameras.

The 1 Mbyte VRAM memory of the MV-1000 may be expanded to 4 Mbytes with the optional MV-1200 memory expansion module. The on-board memory is organized as a continuous memory array, that can accept up to 8 K pixels per line.

The PCI bus used by the MuTech M-Vision 1000 has a number of distinct architectural advantages that benefit video image capture when working at high frame rates, high spatial resolution, or high color resolution. These advantages are most clearly understood when contrasted to the alternative PC bus solutions — the ISA bus and VL-Bus.

The M-Vision 1000 has been designed to sustain maximum PCI bus throughput, which on a suitably equipped PC could reach 50 Mbytes per second. And, unlike the VL-Bus, it can do so without contending with the system processor or other high speed peripherals, such as SCSI disk controllers. A final benefit for system integrators using the M-Vision 1000 is that it is truly platform independent, and is compatible with computers using not only Intel, but also Power PC, and Digital Alpha processors.

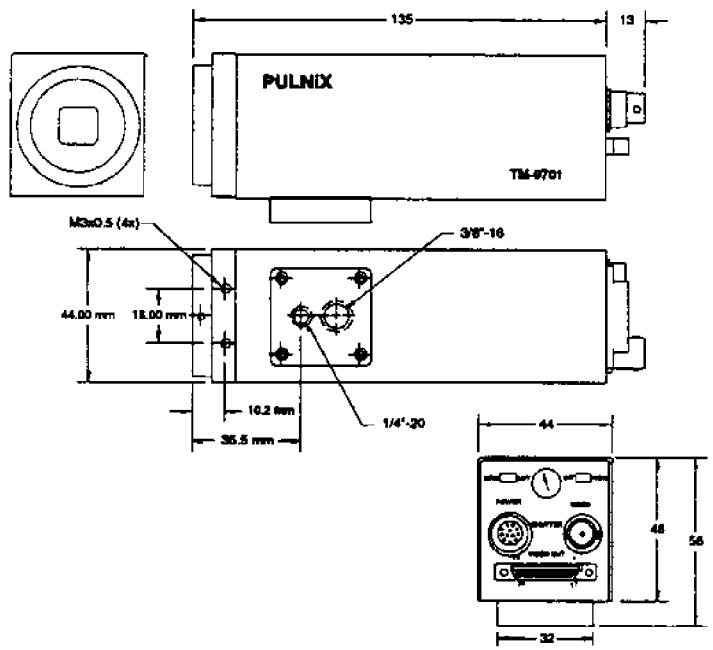
MuTech Corp • 85 Rangeway Road • Billerica, MA 01862 • Tel. 508-663-2400 • Fax. 508-663-3444

Specifications

Model	TM-9701
Imager	2/3" progressive scanning interline transfer CCD
Pixel	768 (H) x 484 (V)
Cell size	11.6 µm x 13.6 µm progressive scan
Scanning	525 lines 30 Hz or 60 Hz 2:1 interlace
Sync	Internal/external auto switch HD/VD, 4.0 Vp-p impedance 4.7KΩ VD=interlace/non-interlace HD=15.734kHz±5%
Data clock output	14.31818 MHz
TV resolution	570 (H) x 484 (V)
S/N ratio	50dB min.(AGC = off)
Min. illumination	1.0 lux, f=1.4 without IR cut filter (no shutter). Sensitivity: 10µV/e-
Video output	1.0 Vp-p composite video, 75Ω and 8-bit RS422 output
AGC	ON/OFF (OFF std.)
Gamma	0.45 or 1.0 (1.0 std.)
Lens mount	C-mount
Power req.	12V DC 500 mA
Operating temp.	-10°C to 50°C
Vibration & shock	Vibration: 7G (200Hz to 2000 Hz) Shock: 70G
Size (W X H X L)	44mm x 48.5mm x 136mm 1.73" x 1.91" x 5.35"
Weight	323 grams (11.4oz)
Power cable	12P-02
Power supply	K25-12V or PD-12
Auto iris connector	None
Functional options	OP-51, OP-29-5
Accessories	30DG-02 digital cable, MP-211-031-113-4300 31-pin mating connector included

Due to ongoing product improvements, specifications may change without notice.

Physical Dimensions



Pin Configuration

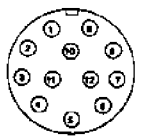
31-Pin connector (MP211-031-113-4300)

Pin#	Description	I/O	Pin#	Description	I/O
1	CLK+	Out	17	CLK-	Out
2	LDV+	Out	18	LDV-	Out
3	FDV+	Out	19	FDV-	Out
4	GND		20	VINIT	In
5	EXT HD	In	21	EXT VD	In
6	INTEG	In	22	ENINT	In
7	LPULSE	Out	23	GND	
8	D0+	Out	24	D0-	Out
9	D1+	Out	25	D1-	Out
10	D2+	Out	26	D2-	Out
11	D3+	Out	27	D3-	Out
12	D4+	Out	28	D4-	Out
13	D5+	Out	29	D5-	Out
14	D6+	Out	30	D6-	Out
15	D7+	Out	31	D7-	Out
16	GND				

Note: CLK: data clock, LDV: Line data valid, FDV: Frame or field data valid, ENINT: Integration enable, LPULSE: Last pulse

12-Pin Connector

1	GND	7	VD in
2	+12V	8	GND
3	GND	9	HD in
4	Video	10	GND
5	GND	11	Int. cont
6	VINIT	12	GND



Shutter Control Switch

	Manual	Async
0	no shutter	no shutter
1	1/60	1/16000
2	1/125	1/8000
3	1/250	1/4000
4	1/500	1/2000
5	1/1000	1/1000
6	1/2000	1/500
7	1/4,000	1/250
8	1/8,000	1/125
9	1/16,000	Double pulse

Distributed By:

JKN Electronics, Inc.
8 High Street, P.O. Box 1031
Bellingham, MA 02019
(508) 966-2721

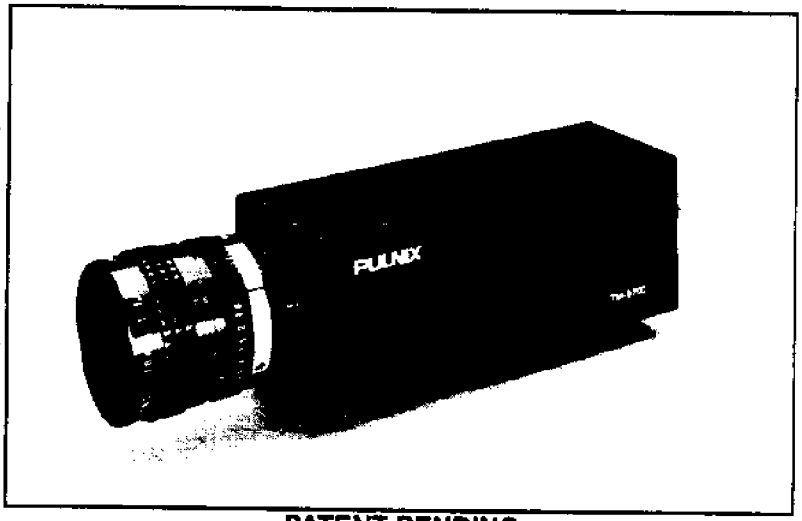
In Australia
PULNiX America Inc.
Unit 16, #35 Garden Road
Clayton, Vic 3168
Tel: 3-9546-0222
Fax: 3-9562-4892

In the
PULN
Aviary
Basing
Tel: 01
Fax: 0

In Japan
Takenaka System Co. Ltd.
7-1, Shinomiya Nanao Cho
Yamashina-ku, Kyoto, 607
Tel: 075-593-9787

VSI/D/970/1995

PULNiX TM-9701 PROGRESSIVE SCANNING FULL FRAME SHUTTER CAMERA



PATENT PENDING

Features

- **Very high resolution 2/3" progressive scanning interline transfer CCD imager 768(H) x 484(V)**
- **Digital output for progressive scan (525 lines) or interlace (RS-170) output with EIA-422 format**
- **Full frame shutter ....1/60 to 1/16,000 sec.**
- **Asynchronous reset with ext. shutter control**
- **Frame memory built-in for async image capturing**
- **Full frame integration with uninterrupted video**
- **Excellent S/N (50 dB)**
- **AGC on/off, gamma 1 or 0.45**
- **Small, light weight, high-rel connectors**

General Description

The PULNiX TM-9701 is a high resolution 768 H x 484 V black and white full frame shutter camera with asynchronous reset capability and uniform MTF (Modulation Transfer Function) characteristics. These cameras are excellent in applications such as bar code reading, on-line inspection, gauging, character reading, high definition graphics, intensified CCD cameras, and detailed surveillance. Added to the wide versatility of this camera is an 8-bit digital signal output through EIA-422. The signal is interlace (RS-170) or progressive scanning (525 lines). It has built-in frame memory for async or integration image capturing without using special frame grabbers.

Asynchronous reset and full frame integration are standard features of these cameras. AGC enable, internal IR cut filter, gamma adjust to 0.45, and the popular remoted imagers (TBD) are all optional features that PULNiX offers for these cameras.

Asynchronous Reset

The TM-9701's asynchronous reset is flexible and takes external HD for phase locking. When VINIT pulse is applied, it resets the camera's scanning and purging of the CCD. There are three modes to control the asynchronous reset and shutter speed:

1. **External VINIT with double pulse** in which the duration between two pulses controls the shutter speed externally.
2. **Internal shutter speed with Fast mode** in which the video signal has no delay from the reset timing (shutter speed range is 1/2000 to 1/16,000 sec.)
3. **Internal shutter speed with Slow mode** which can vary the speed control from 1/60 to 1/1,000 sec. The video signal starts with internal V reset timing related to shutter speed.

The built-in frame memory can maintain the asynchronously captured full frame image until next VINIT pulse comes in. The output can be either interlace or progressive scanning. Both analog RS-170 format (1Vp-p, 75Ω) and 8 bit digital format (EIA-422) are available from the camera.

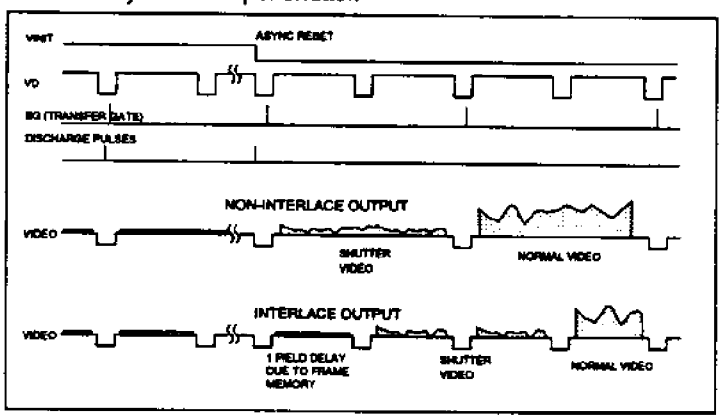
Integration

The CCD imager of the TM-9701 can be exposed longer than normal TV timing (1/30 sec.). This feature provides high sensitivity for dark environment applications. Integration is achieved by controlling the #11 pin of the 12-pin connector to Low (GND). Because of the progressive scanning CCD chip in the TM-9701, a full frame of resolution is obtainable with either interlace or progressive scan format. The internal frame memory provides continuous video output without frame grabber.

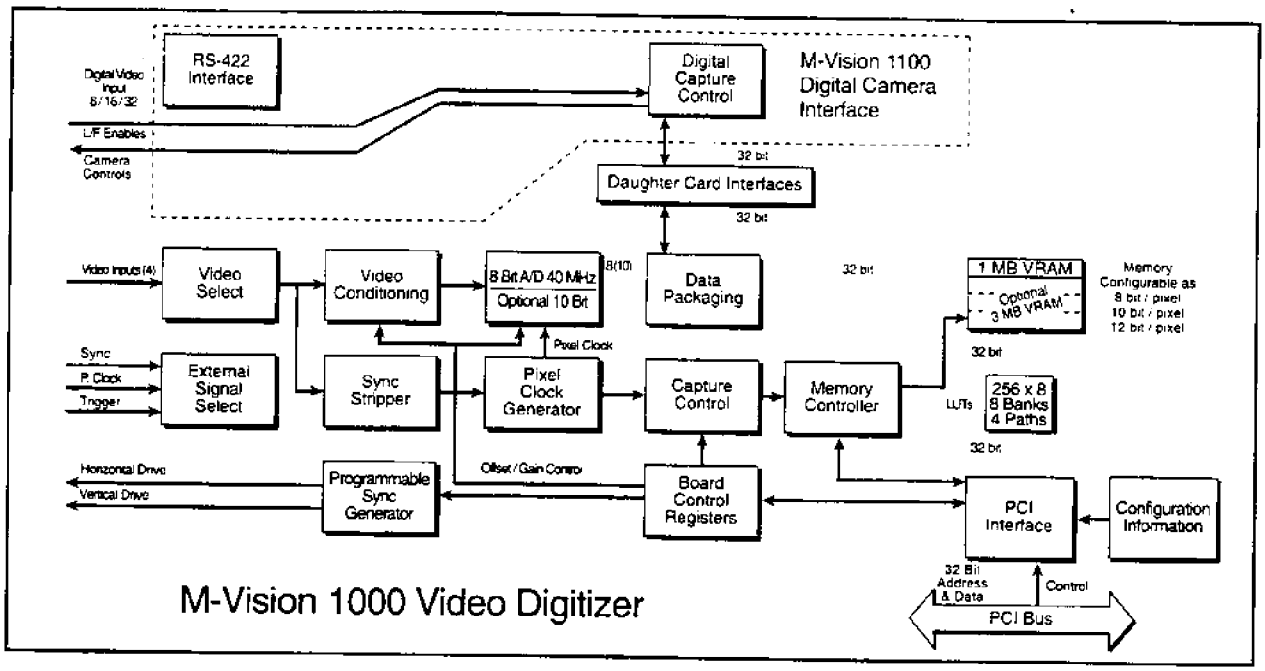
Electronic Shutter

The TM-9701 has a substrate drain type shutter mechanism which provides a superb picture at various speeds without smearing. Built-in manual shutter speed control varies the electronic shutter rate between 1/60, 1/125, 1/250, 1/500, 1/1000, 1/2000, 1/4000, 1/8,000, and 1/16,000 sec.

With VINIT high (5V), the CCD keeps discharging; with a negative going pulse to VINIT, the camera resets and purges the charge momentarily, then it starts integrating for the period of shutter control set by either an external double pulse or internal shutter control. Because of the progressive scanning, a full 484 lines of vertical resolution per single shutter is available unlike a normal CCD camera which is only 244 lines per shutter.



HARDWARE SPECIFICATIONS



M-VISION 1000 (MV-1000)

Analog Video Digitization

- 4:1 MUX selects analog input
- RS-170/CCIR
- AC/DC Coupled Input
- 8 bit A/D programmable up to 40 MHz
- Accepts Composite or Separate H/V Sync
- Ext. Trigger/Ext. Pixel Clock (to 40 MHz)
- 10 bit A/D (optional)
- Differential Input
- Up to 8K Pixels per Line
- Up to 8K Lines/Frame

Frame Buffer

- 1 Mbyte fast page VRAM
- Dual ported memory enables 50 Mbytes/ second access
- Simultaneous host access
- Entire on board memory mapped into 4 GB PCI address
- Supports PCI 8/16/32 bit accessing
- Mask Register for bit plane protection
- Can be configured as 8/10/12 bit frame buffer
- 8 x 256 x 8 LUTs (configurable as 10 to 8 bit)

Phase-Lock-Loop & A/D Control

- Less than 10ns (typically less than 5ns) jitter
- 5-40 MHz programmable
- 12 bit PLL counter
- Square Pixel acquisition/programmable to other aspect ratios
- Digitize Frame or Field (odd, even, next)
- Digitize Progressive Frame
- Sub-sample X2 and X4
- Digitize Frame Sequence
- Digitizing window of the video is fully programmable

Video Signal Conditioning

- Programmable Band Limiting Filter
- Clamping on back porch or sync tip (enable/disable)
- High/Low digitizing range programmable (Gain/Offset)

Other Controls

- All Controls registers are PCI memory mapped
- Status Register (V. Sync., Field, Ext. Trigger)
- Programmable interupt request
- General Purpose Control Signals (e.g. Asynchronous reset)
- Master and Slave mode Data Transfers

DIGITAL CONTROL INTERFACE (MV-1100)

- Supports EIA 422 Standard
- 8/16/32 bit differential input digital video 32 bit @ 110 MHz 8 bit input @ 40 MHz
- Inputs Supported
  - Line enable
  - Field/Frame Enable
  - Field Flag
  - P Clock < 40 MHz
- Outputs Available
  - General Purpose Control (e.g. for strobes)
  - P Clock
  - H Drive
  - V Drive

MEMORY UPGRADE MODULE (MV-1200)

- 3 Mbyte VRAM memory Module

Distributed By:

JKN Electronics, Inc.
8 High Street, P.O. Box 1031
Bellingham, MA 02019
(508) 966-2721

TECH
MuTech Corporation

85 Rangeway Road
Billerica, MA 01862
Tel. 508-663-2400 • Fax. 508-663-3444

Appendix E: Decision Matrices

Decision Matrix for Pulnix TM-9701AN Camera/Monochrome Analog Output										
Criteria	**Bus Master**	**On Board Memory**	**Output Format**	**Periferal Control**	**Support**	**Software**	**Adaptability**	**Compression**	**Price**	**Total**
Value	9	3	7	4	6	7	4	5	20	65
	0.14	0.05	0.11	0.06	0.09	0.11	0.06	0.08	0.31	**1.00**
MV-1000	10	10	10	10	9	9	10	0	7	
$1,475	1.38	0.46	1.08	0.62	0.83	0.97	0.62	0.00	2.15	**8.11**
IMAQ PCI-1408	7	2	10	8	5	10	10	0	9	
$1,195	0.97	0.09	1.08	0.49	0.46	1.08	0.62	0.00	2.77	**7.55**
Oculus-TCI	7	10	6	10	10	6	10	0	5	
$1,790	0.97	0.46	0.65	0.62	0.92	0.65	0.62	0.00	1.54	**6.42**
Falcon-PCI	10	10	10	10	10	10	10	0	3	
$3,500	1.38	0.46	1.08	0.62	0.92	1.08	0.62	0.00	0.92	**7.08**
FPG-44	7	10	10	10	10	10	10	0	6	
$2,145	0.97	0.46	1.08	0.62	0.92	1.08	0.62	0.00	1.85	**7.58**
Data Raptor	10	10	10	10	10	10	10	0	3	
$3,000	1.38	0.46	1.08	0.62	0.92	1.08	0.62	0.00	0.92	**7.08**
XPG-1000	10	10	10	10	10	10	10	0	3	
$3,800	1.38	0.46	1.08	0.62	0.92	1.08	0.62	0.00	0.92	**7.08**

Decision Matrix for Pulnix TM-9701AN Camera/Monochrome Digital Output										
Criteria	Bus Master	On Board Memory	Output Format	Periferal Control	Support	Software	Adaptability	Compression	Price	Total
Value	9	3	7	4	6	7	4	5	20	65
	0.14	0.05	0.11	0.06	0.09	0.11	0.06	0.08	0.31	
MV-1000	10	10	10	10	10	9	10	0	8	
$2,140	1.38	0.46	1.08	0.62	0.92	0.97	0.62	0.00	2.46	**8.51**
Falcon-PCI	10	10	10	10	10	10	10	0	4	
$3,500	1.38	0.46	1.08	0.62	0.92	1.08	0.62	0.00	1.23	**7.38**
Road Runner	10	0	10	10	10	10	10	0	9	
$2,100	1.38	0.00	1.08	0.62	0.92	1.08	0.62	0.00	2.77	**8.46**
FPG-44	10	10	10	10	10	10	10	0	8	
$2,145	1.38	0.46	1.08	0.62	0.92	1.08	0.62	0.00	2.46	**8.62**
Data Raptor-VL	10	10	10	10	10	10	10	0	6	
$3,000	1.38	0.46	1.08	0.62	0.92	1.08	0.62	0.00	1.85	**8.00**
XPG-1000	10	10	10	10	10	10	10	10	3	
$3,800	1.38	0.46	1.08	0.62	0.92	1.08	0.62	0.80	0.92	**7.88**