Detection of a Virtual Passive Pointer

Naren Vira nvira@howard.edu
Professor
Department of Mechanical Engineering
Howard University
Washington, DC 20059, USA

Shaleen Vira svira@nyu.edu
Medical Student
School of Medicine
New York University
New York, NY 10016, USA

ABSTRACT

The paper presents a methodology for detecting a virtual passive pointer. The passive pointer or device does not have any active energy source within it (as opposed to a laser pointer) and thus cannot easily be detected or identified. The modeling and simulation task is carried out by generating high resolution color images of a pointer viewing via two digital cameras with a popular three-dimensional (3D) computer graphics and animation program, Studio 3D Max by Discreet. These images are then retrieved for analysis into a Microsoft's Visual C^{++} program developed based on the theory of image triangulation. The program outputs a precise coordinates of the pointer in the 3D space in addition to it's projection on a view screen located in a large display/presentation room. The computational results of the pointer projection are compared with the known locations specified by the Studio 3D Max for different simulated configurations. High pointing accuracy is achieved: a pointer kept 30 feet away correctly hits the target location within a few inches. Thus this technology can be used in presenter-audience applications.

Keywords: Modeling and Simulation, Image Processing, Triangulation Technique, Virtual Pointer Detection, Interactive Large Display

1. INTRODUCTION

Pointing devices, namely laser pointers, are commonly used to indicate a specific position on a viewing screen to an audience. Laser pointers utilize an active energy source, a concentrated photon/energy beam that streams from the device to the nearest physical object, hopefully the slide/screen. Occasionally, accidental pointing is hazardous. This work demonstrates the use of a passive device, one that does not require any energy source. However, external detecting mechanisms to precisely identify where the pointer is pointing to are required. To achieve this requisite, two high resolution color cameras and image triangulation methodology for pointer detection analysis were employed.

Another limitation of laser pointers is that every audience member does not have one (or carries it around to every meeting!) and thus resort to hand gestures along with verbal cues ("No, not there, over there") to instruct the presenter where to look when asking a question pertaining to a specific point in a slide/view screen. This ineffective communication method is exacerbated in a large room with a large audience. Similarly, a presenter pointing to information on the display by hand during the presentation cannot clearly be visualized and understood by the audience. The proposed technique overcomes these difficulties by allowing both parties to interact simultaneously with the use of many inexpensive passive pointers. And these multiple pointers can be tracked with the use of two cameras that view the room containing the audience and presenter. The monitoring computer outputs different color dots on the view screen for precise pointing direction by either party, resulting in intelligent and communicable interaction between audience and presenter.

Figure 1a: Airport Control Center

Figure 1b: Air Traffic Simulator at Command and Control Center

Furthermore, the long term thrust of the work is to explore gesture recognition technology applicable for an interactive large display environment (see Section 1.1). The method of tracking a passive pointer can easily be adopted for its use in gesture detection and control. Obviously the gesture grammar, a set of rules on which the gestures are interpreted, needs to be incorporated (refer Section 1.2 for details on gesture recognition). This work is also a stepping stone for

developing an intelligent non-touch computer screen interface. Let us visualize two web cameras mounted on top of a computer screen viewing the computer user. The camera can track a non-touch passive pointer or user's finger as it approaches the screen. Once the camera and associated interface identify the pointing location on the screen, it can zoom in or out showing details as the finger, respectively, move towards or away from the screen. A simple example would be to view a geographical map with zooming in and out capability. The interface can also pop out or display additional details/information, if needed, in another window of the pointing location. The example for this scenario would be its use in a tower simulator when a controller points at an aircraft; the necessary detail information regarding aircraft can appear on the large screen display, a personalized small screen display, or even on their own head worn display. The output information can thus be quickly accessed and utilized.

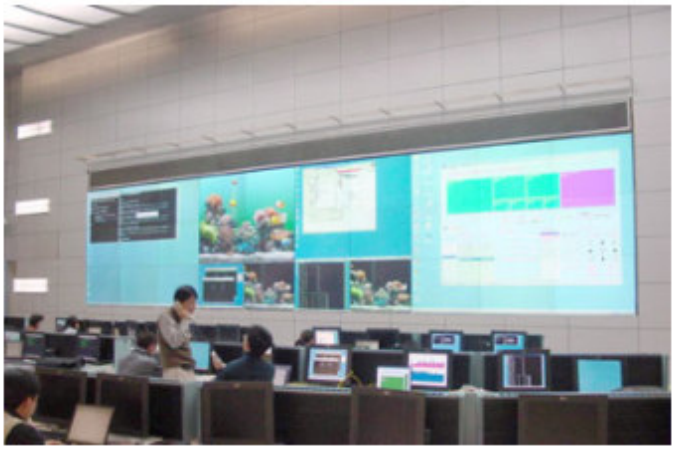

Figure 1c: Commercial Bank

Figure 1d: Interactive DataWall of AFRL, US Air Force

1.1 Commercial Application

This section briefly outlines the use of passive pointers in a large room setting. Figure 1 shows a few potential uses of adopting the present work: airport control centers, air traffic simulators, bank centers and the US Air Force interactive DataWall. These settings represent a large display area to address the problem of information management in the 21st century. One can also incorporate several multimedia capabilities such as audio/visual, conventional computing, tractable laser pointers and wireless interactive technology. For additional information on interactive DataWall of

AFRL (multimedia display with combined resolution of 3840 x 1024 pixel across a 12' x 3' screen area), refer to the web pointer presented in Reference [1].

1.2 Gesture Recognition Technology

Hand gestures provide a useful interface for humans to interact with not only other humans but also machines. For high degree-of-freedom manipulation tasks such as the operation of three-dimensional (3D) objects in virtual scenes, the traditional interface composed of a keyboard and mouse is neither intuitive nor easy to operate. For such a task, we consider direct manipulation with hand gestures as an alternative method. This would allow a user to directly indicate 3D points and issue manipulation commands with his/her own hand.

In the past, this idea has led to many gesture-based systems using glove-type sensing devices in the early days of virtual reality research. Such contact-type devices, however, are troublesome to put on and take off, and continuous use results in fatigue. To overcome these disadvantages, vision researchers tried to develop non-contact type systems to direct human hand motion [2, 3, 4]. These works had some instability problems particular to vision based systems. The most significant problem is occlusion: vision systems conventionally require match of detected feature points between images to reconstructed 3D information. However, for moving non-rigid objects like a human hand, detection and matching of feature points is difficult to accomplish correctly.

Providing a computer with the ability to interpret human hand is a step toward more natural human-machine interactions. Existing input systems augmented with this, as well as such other human-like modalities such as speech recognition and facial expression understanding, will add a powerful new dimension to the range of future, as well as current, computer applications. A wide spectrum of research is underway on the problem of gesture interpretation. The primary reason for the advancement is the continuously declining expenses of hardware and image grabbing and processing. Even color processing is now available and it is fast enough for pattern recognition.

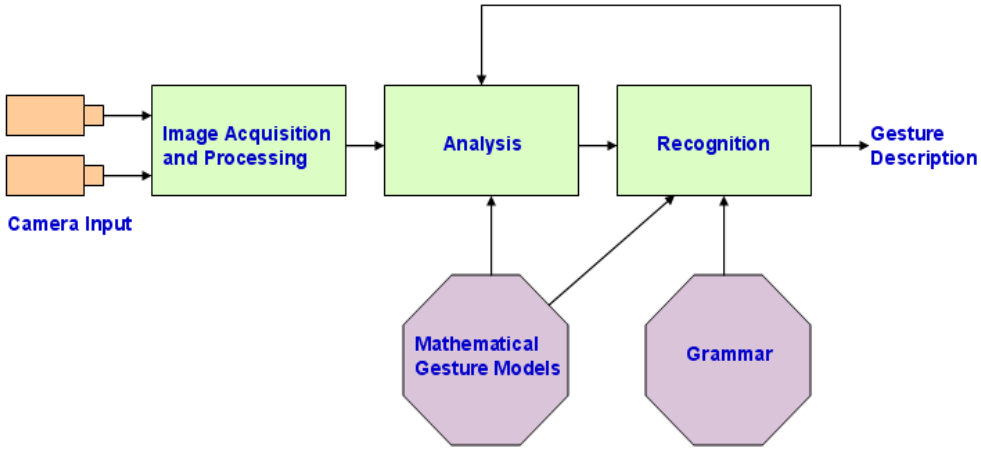

Figure 2: Gesture Interpretation System

Currently there is no universal definition of what a gesture recognition system should do or even what is a gesture. Our definition of gesture from the perspective of a computer is simply a temporal sequence of hand images. An element from a finite set of static hand poses is the expected content with an image frame. A gesture is, therefore, a sequence of static hand poses. Poses are assumed to contain the identity of the hand shape and (possibly) the orientation, translation and distance from camera information. The spatio-temporal nature of the gesture data make the gesture state unmeasurable at a given instance in time, but for each time step we can determine the static hand pose. A general gesture recognition system is depicted in Figure 2. Visual images of gestures are acquired by one or more cameras. They are processed in the

analysis stage where the gesture model parameters are estimated. Using the estimated parameters and some higher level knowledge, the observed gestures are inferred in the recognition stage. The grammar provides a set of rules on which the gestures are interpreted.

2. METHODOLOGY

This section outlines the modeling theory considered for detecting the passive pointer.

2.1 Camera and Image Processing

Consider a system with two cameras of focal length f and baseline distance b as shown in Figure 3. The optical axes of the two cameras are converging with an angle θ and that all geometrical parameters (b, f, and θ) are a priori known or estimated using a camera calibration technique Refs. [5 - 8]. A feature in the scene depicted at the point P is viewed by the two cameras at different positions in the image planes (I_1 and I_2). The origins of each camera's coordinate system are located at the camera's center which is at a distance f away from the corresponding image planes I_1 and I_2, respectively. It is assumed, without loss of generality, that the world coordinate system (Cartesian coordinates X, Y, and Z) coincides with the coordinate system of camera 1 (left camera), while the coordinate system of camera 2 (right camera) is obtained from the former through rotation and translations.

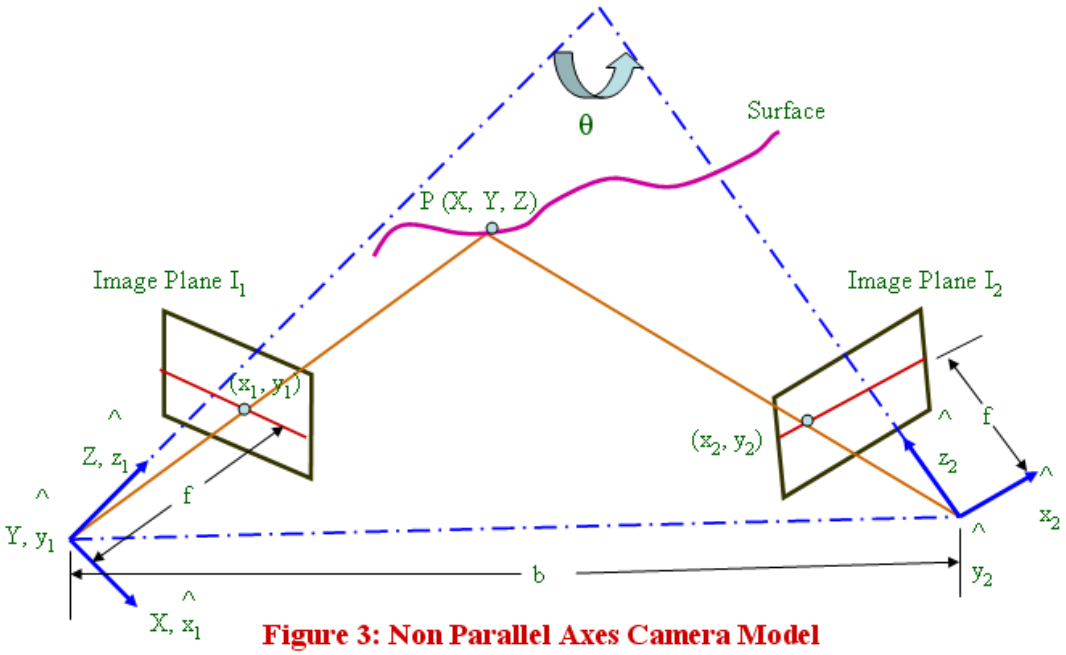

Figure 3: Non Parallel Axes Camera Model

The plane passing through the camera centers and the feature point in the scene is called the epipolar plane. The intersection of the epipolar plane with the image plane defines the epipolar line as shown in Figure 4. For the model shown in the figure, every feature in one image will lie on the same row in the second image. In practice, there may be a vertical disparity due to misregistration of the epipolar lines. Many formulations of binocular stereo algorithms assume zero vertical disparity.

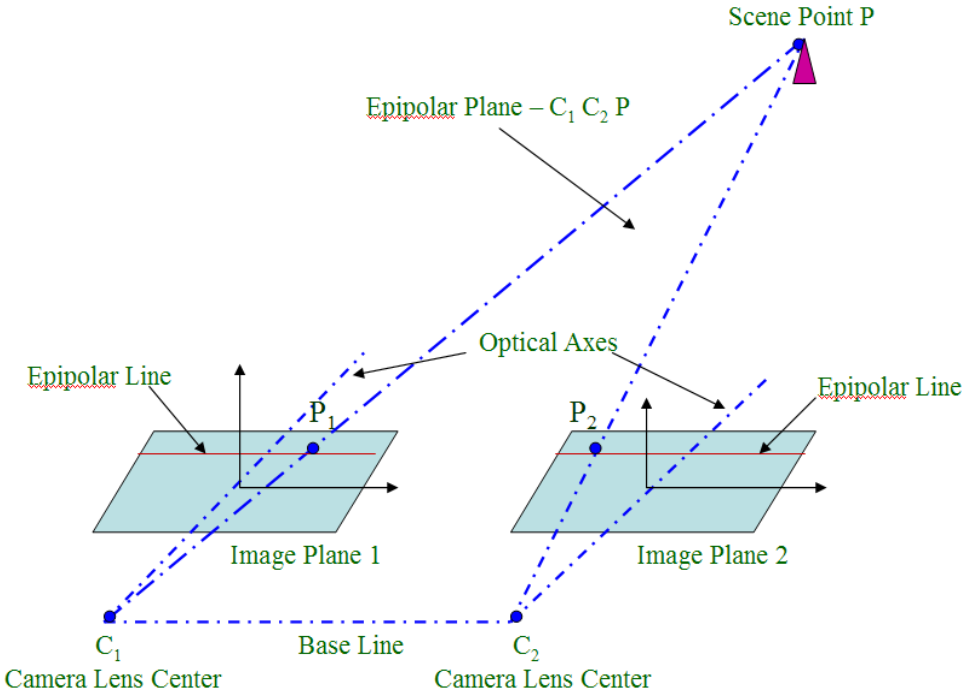

Figure 4: The Epipolar Plane

As illustrated in Figure 3, point P with the world coordinates (X, Y, and Z) is projected on image plane I_l as point (x_1, y_1) and image plane I_2 as point (x_2, y_2). Then, assuming a perspective projection scheme, a simple relation between the left camera coordinates (x_1, y_1) and world coordinates (X, Y, and Z) can be written as

$$x_1 = f * X / Z \quad \text{and} \quad y_1 = f * Y / Z \tag{1}$$

Similarly, we can write for right camera as

$$x_2 = f * x_2^{\wedge} / z_2^{\wedge} \quad \text{and} \quad y_2 = f * y_2^{\wedge} / z_2^{\wedge} \tag{2}$$

Where, coordinate system of camera 2 ($x_2^{\wedge}$, $y_2^{\wedge}$ and $z_2^{\wedge}$) is related with respect to the world coordinate system by simply translation and rotation as

$$\begin{aligned} x_2^{\wedge} &= c\,X + s\,Z - b\,c' \\ y_2^{\wedge} &= Y \\ z_2^{\wedge} &= -s\,X + c\,Z + b\,s' \end{aligned} \tag{3}$$

Here, symbols c = cos (θ) and s = sin (θ), c' = cos (θ/2) and s' = sin (θ/2) are used. Substituting Eq. (3) into Eq. (2), we can write

$$\begin{aligned} x_2 &= f\,[(c\,X + s\,Z - b\,c') / (-s\,X + c\,Z + b\,s')] \\ y_2 &= f\,[Y/(-s\,X + c\,Z + b\,s')] \end{aligned} \tag{4}$$

Combining Eq. (1) and Eq. (4), lead to

$$\begin{aligned} x_2 &= f\,[(f\,s + x_1\,c)Z - f\,b\,c'] / [(f\,c - x_1\,s)Z + f\,b\,s'] \\ y_2 &= (f\,Z\,y_1) / [(f\,c - x_1\,s)Z + f\,b\,s'] \end{aligned} \tag{5}$$

It can be observed for Eq. (5) that the depth Z of the scene point P can be estimated if its projections (x_1, y_1) and (x_2, y_2) on image planes I_1 and I_2, respectively, are known. That is for a given point (x_1, y_1) on I_1, its corresponding point (x_2, y_2) on I_2 should be found. Hence, we can define a disparity vector $\mathbf{d} = [d_x, d_y]^T$ at location (x_2, y_2) of camera 2 with respect to camera 1 as

$$d_x = x_1 - x_2 \tag{6}$$

$$= \frac{f\,b\,(f\,c' + x_1\,s') + [\;x_1(f\,c - x_1 s) - f\,(f\,s + x_1 c)\;]Z}{(f\,c - x_1\,s)Z + f\,b\,s'}$$

$$d_y = y_1 - y_2 \tag{7}$$

$$= \frac{f\,b\,s'\,y_1 + [\;(f\,c - x_1\,s)\;- f\;]\,y_1\,Z}{(f\,c - x_1\,s)Z + f\,b\,s'}$$

When the disparity vector **d** is known, equations 6 and 7 reduce to an over determined linear system of two equations with a single unknown, Z (the depth) and a least-squares solution can be obtained [9]. When cameras axes are parallel (i.e., $\theta = 0$) these equations (Equations (6-7)) can be simplified to (see Reference [10] and Figure 5)

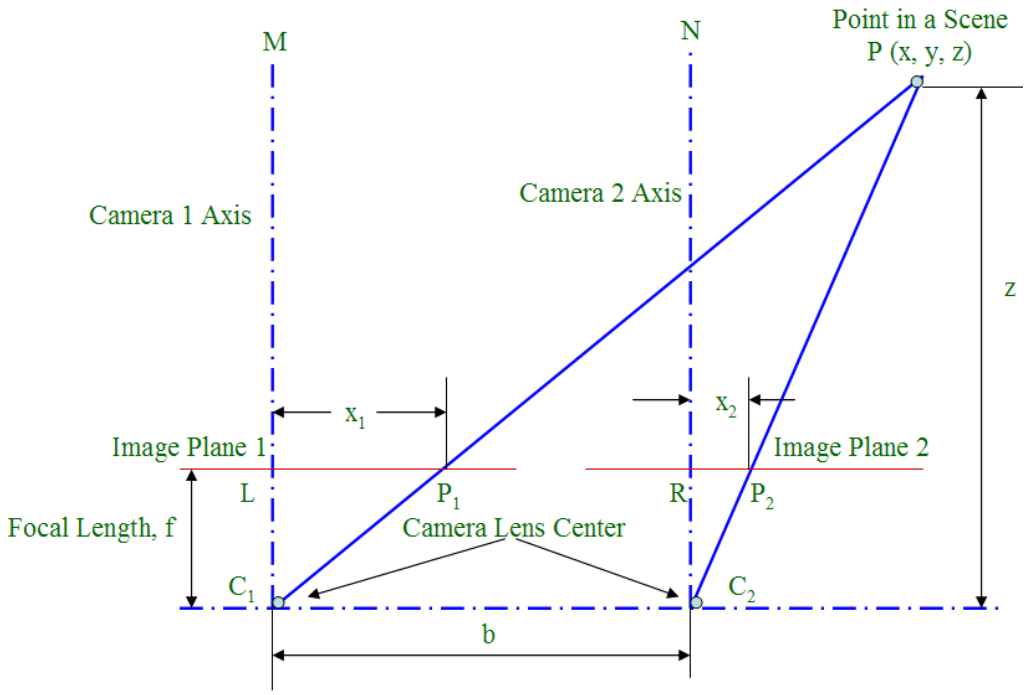

Figure 5: The Parallel Axes Camera Model

$$d_x = f\;\; b / Z \text{ and } \;\; d_y = 0 \tag{8}$$

Thus, the depth at various scene points may be recovered by knowing disparities of corresponding image points.

2.2 Color Representation and Pointer Modeling

The intensity of each image pixel in RGB color space can be represented in a vector form as

$$\mathbf{P}(I, J) = R(I, J)\,\mathbf{e_1} + G(I, J)\,\mathbf{e_2} + B(I, J)\,\mathbf{e_3} \quad (9)$$

The symbol I and J stand for pixel coordinates, and $\mathbf{e_1}$, $\mathbf{e_2}$ and $\mathbf{e_3}$ are unit vectors along R, G, and B color space, respectively. The terms R (I, J), G (I, J), and B (I, J), respectively, represent red, green and blue color intensities. As opposed to stereo matching algorithm (correspondence of every image pixel is found), here we are only interested in identifying those pixels that corresponds to the pointer in one image and respective matching pixels in another image viewed from a second camera. More precisely, if we marked the pointer's ends with two distinct colors then only those pixels are required to be matched in both images. Without loss of generality, let us say that one end is marked with red color and other is with blue. Because we are only interested in matching the pointer's red or blue color end pixels of each image, Equation (9) can be rewritten as

$$\mathbf{P}(I, J) = R(I, J) \quad (10)$$

for the red color end pixels and

$$\mathbf{P}(I, J) = B(I, J) \quad (11)$$

for the blue color end pixels. Alternatively, we scan the whole image to identify all pixel-coordinates I and J that represent either red or blue color end of the pointer. From this information, we compute the centroid of each color end. That is $\mathbf{P_1}$ (I, J) centroid for the red color end as shown in Figure 6, image 1 (left), we have

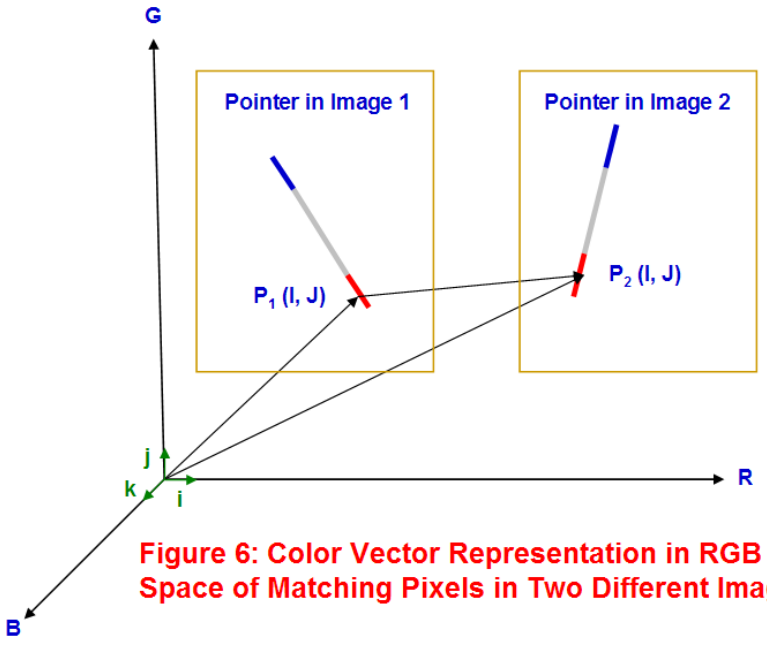

Figure 6: Color Vector Representation in RGB Space of Matching Pixels in Two Different Images

$$\mathbf{P_1}(I, J) = R_1(Imid, Jmid) \quad (12)$$

Where,

Imid = Imin + (Imax – Imin)/2
Jmid = Jmin + (Jmax – Jmin)/2

The terms mid, min, and max correspond to the mid point, minimum location, and maximum location of the color within that particular color end. Note that the image has to be searched to find the min and max locations. The term centroid and mid point of the color end are interchangeable because of two dimensional coordinate system representation. Furthermore, the pointer size is very small in comparison to image size. Similarly, we can compute the centroid of the red color end in image 2 (right) as

$$\mathbf{P_2}\ (x, y) = R_2\ (Imid, Jmid) \qquad (13)$$

We assume that the centroid points $\mathbf{P_1}$ (I, J) and $\mathbf{P_2}$ (I, J) represent the matching points. This assumption is valid because the pointer dimensions are relatively small with respect to the physical dimension of the room (note the image size). Thus, the implication is that the process of disparity analysis needed for stereo matching is not required and the task of finding matching pixels is considerably simplified. The same analysis can be applied for finding the matching points corresponding to the blue color end of the pointer. It should be emphasized that we deliberately chosen two distinct color-ends to simplify and speed up the process of image scanning. One can choose other pixel matching methods depending upon the application.

By knowing the x and y coordinates of each centroid point (after correlating image pixels with the space coordinates) of the pointer in a single image, we can mathematically pass a line through these two points to describe a pointer in a 2D space. Now the process of triangulation in required to compute the three-dimensional coordinates of the pointer from these two images (i.e., from four centroid points).

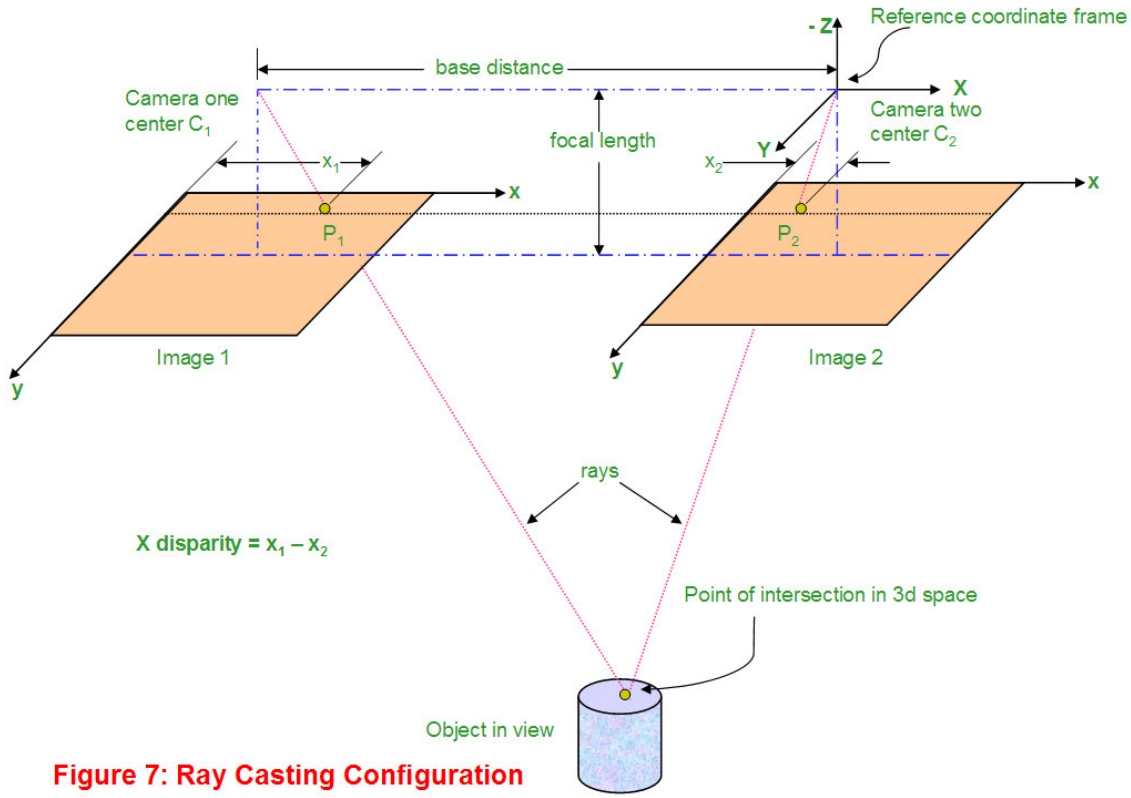

Figure 7: Ray Casting Configuration

2.3 Three-dimensional Triangulation Technique

We apply ray casting analysis to triangulate three-dimensional coordinates of each image pixel point in a space as it is viewed by two cameras with respect to a chosen reference frame. Without loss of generality, the reference fame could be at one of the cameras' center. We have chosen the center location of camera 2 as the frame of reference. Each ray is cast from the viewpoint (in this case, center of the camera) through each pixel of the projection plane (in this case, image planes 1 and 2) into the volume dataset. The two rays wherever they intersect in a 3D space determine the coordinates of a point viewed by both cameras as shown in Figure 7. By connecting all intersecting points in the volume dataset, we can generate a 3D point cloud floating in a space. We utilize four points at a time (two in each image) to compute the three-dimensional coordinates of the pointer's end. Thus, the location of the pointer can be identified in a 3D space from the knowledge of its two ends.

2.3.1 Two Intersecting Line Problem

The computation of a common point from two rays reduces to a problem of two-line intersection each radiating from the center of a camera. The ray line is generated by two points in each image as shown in Figure 8. One point on the line is defined by the camera center and the second point by the centroidal pixel of the pointer end in the image plane (i.e., P_1 or P_2 in Figures 6, 7 and 8). Note that P_1 and P_2 are image matched points.

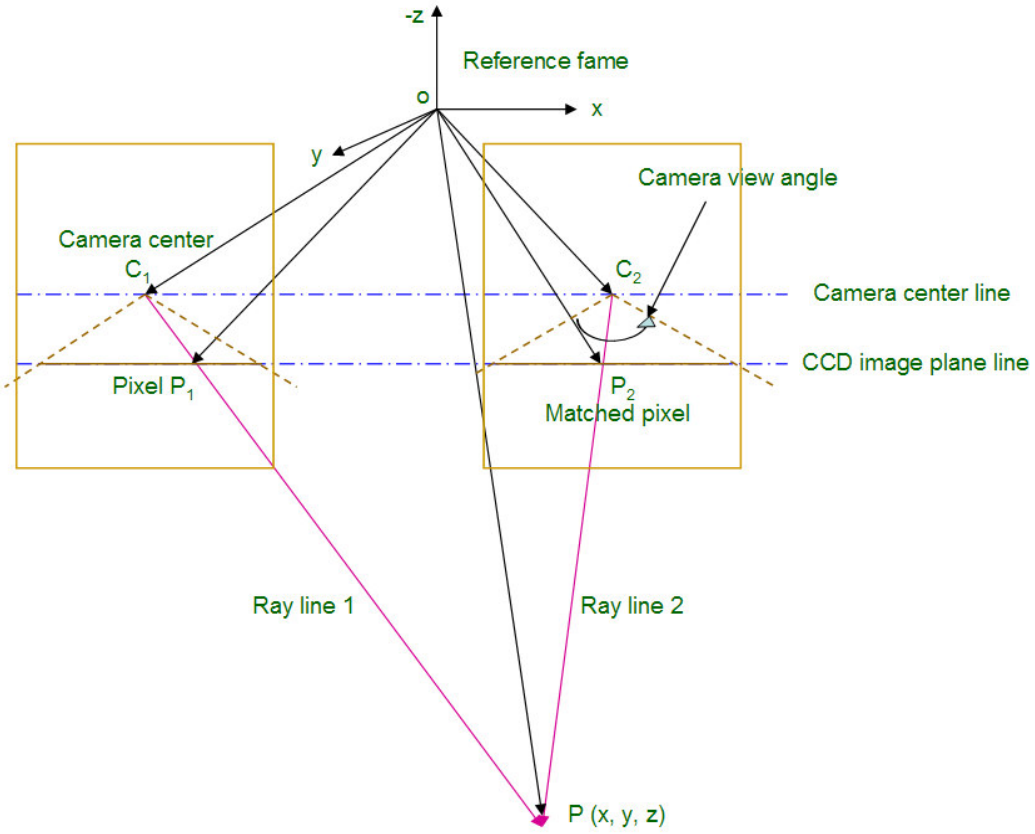

Figure 8: Coordinate Computation for Two Lines of Intersection

Considering a frame of reference (x, y, z), the point sets (C_1, P_1) and (C_2, P_2) are situated on the ray lines 1 and 2, respectively, as depicted in Figure 8. Since the points P_1 (I, J) and P_2 (I, J) are identified by the pixel coordinates, they need to be converted into the physical space. Thus, the following linear space transformation is used:

$$\text{x distance per pixel} = \frac{f * \tan(\text{half view angle of camera})}{(\text{Image width in pixel}) / 2} \quad (14)$$

Similarly, y distance per pixel can be correlated. Note that f denotes camera focal length. Because we are interested in computing coordinates of the common point P, let us define each point on the line in x, y, and z reference frames as

$$\mathbf{P} = x\,\mathbf{i} + y\,\mathbf{j} + z\,\mathbf{k} \quad (15)$$
$$\mathbf{P_1} = P_{x1}\,\mathbf{i} + P_{y1}\,\mathbf{j} + P_{z1}\,\mathbf{k}$$
$$\mathbf{P_2} = P_{x2}\,\mathbf{i} + P_{y2}\,\mathbf{j} + P_{z2}\,\mathbf{k}$$
$$\mathbf{C_1} = C_{x1}\,\mathbf{i} + C_{y1}\,\mathbf{j} + C_{z1}\,\mathbf{k}$$
$$\mathbf{C_2} = C_{x2}\,\mathbf{i} + C_{y2}\,\mathbf{j} + C_{z2}\,\mathbf{k}$$

Where **i**, **j**, and **k** are unit vectors along x, y and z axes, respectively. With the condition that the four points must be coplanar (when the lines are not skewed), we can write

$$(\mathbf{C_2} - \mathbf{C_1}) \bullet [(\mathbf{P_1} - \mathbf{C_1}) \times (\mathbf{P_2} - \mathbf{C_2})] = 0 \quad (16)$$

where symbols "•" and "x" represent vector dot and cross product respectively. If s and t are scalar quantities then the common point **P** can be expressed parametrically as

$$\mathbf{P} = \mathbf{C_1} + s\,(\mathbf{P_1} - \mathbf{C_1}) = \mathbf{C_1} + s\,\mathbf{A} \quad (17a)$$
$$\mathbf{P} = \mathbf{C_2} + t\,(\mathbf{P_2} - \mathbf{C_2}) = \mathbf{C_2} + t\,\mathbf{B} \quad (17b)$$

Simultaneous solution of equations (17a) and (17b) yields the value of s as

$$s = \frac{[(\mathbf{C_2} - \mathbf{C_1}) \times \mathbf{B})] \bullet (\mathbf{A} \times \mathbf{B})}{|\,\mathbf{A} \times \mathbf{B}\,|^2} \quad (17c)$$

2.4 Accounting for Camera's Rotations

Six degrees-of-freedom are required to uniquely describe a point in three-dimensional space. One can choose three linear and three rotational coordinate axes. Determination of the pointer's position defined by three linear coordinates (x, y, z) is presented above, whereas orientation of the pointer specified by three rotations (θ, φ, ψ) is given in this section. Thus the rotational motion of the camera is accounted for by the pointer's position and orientation analysis. Define camera's each axis of rotation as pitch, yaw and roll along x, y and z axes, respectively, as depicted in Figure 9. Hence, each axis transformation is given by

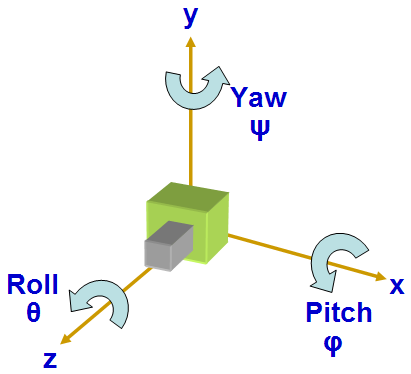

Figure 9: Camera's Pitch Yaw and Roll Axes

$$R(x, \text{pitch}) = R(x, \varphi) = \begin{Bmatrix} 1 & 0 & 0 \\ 0 & C\varphi & -S\varphi \\ 0 & S\varphi & C\varphi \end{Bmatrix} \quad (18)$$

$$R(y, \text{yaw}) = R(y, \psi) = \begin{Bmatrix} C\psi & 0 & S\psi \\ 0 & 1 & 0 \\ -S\psi & 0 & C\psi \end{Bmatrix} \quad (19)$$

$$R(z, \text{roll}) = R(z, \theta) = \begin{Bmatrix} C\theta & -S\theta & 0 \\ S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{Bmatrix} \quad (20)$$

Where, the notations S(angle) = sin (angle) and C(angle) = cos (angle) are used. The combined transformation pitch-yaw-roll can be written as PYR

$$\begin{aligned} PYR &= R(x, \text{pitch})\, R(y, \text{yaw})\, R(z, \text{roll}) \\ &= R(x, \varphi)\, R(y, \psi)\, R(z, \theta) \end{aligned} \quad (21)$$

$$= \begin{Bmatrix} C\theta\, C\psi & -S\theta\, C\psi & S\psi \\ C\theta\, S\varphi S\psi + C\varphi\, S\theta & C\theta\, C\varphi - S\theta\, S\varphi\, S\psi & -C\psi\, S\varphi \\ S\theta\, S\varphi - C\theta\, C\varphi\, S\psi & S\theta C\varphi\, S\psi + C\theta S\varphi & C\varphi\, C\psi \end{Bmatrix}$$

The world coordinates (x, y, z) are, thus, related to camera's view coordinates (x', y', z') as

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \{PYR\} \begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} \qquad \begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = \{PYR^{-1}\} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} \quad (22)$$

Note that inverse transformation is used to account for camera rotations.

2.5 Point of Projection on a View Screen

Knowing the three-dimensional coordinates of common point corresponding to each end of the pointer (after triangulation of red and blue centroids), we can represent the pointer in a 3D space by a line passing through these two points. Figure 10 depicts the pointer connecting red and blue centroidal points $\mathbf{P_r}$ and $\mathbf{P_b}$ respectively. The projection of this line on a plane described by the view screen is of our interest. Thus, the problem is now simplified to finding coordinates of intersecting point between line and a plane as shown by point $\mathbf{P_i}$ in Figure 10.

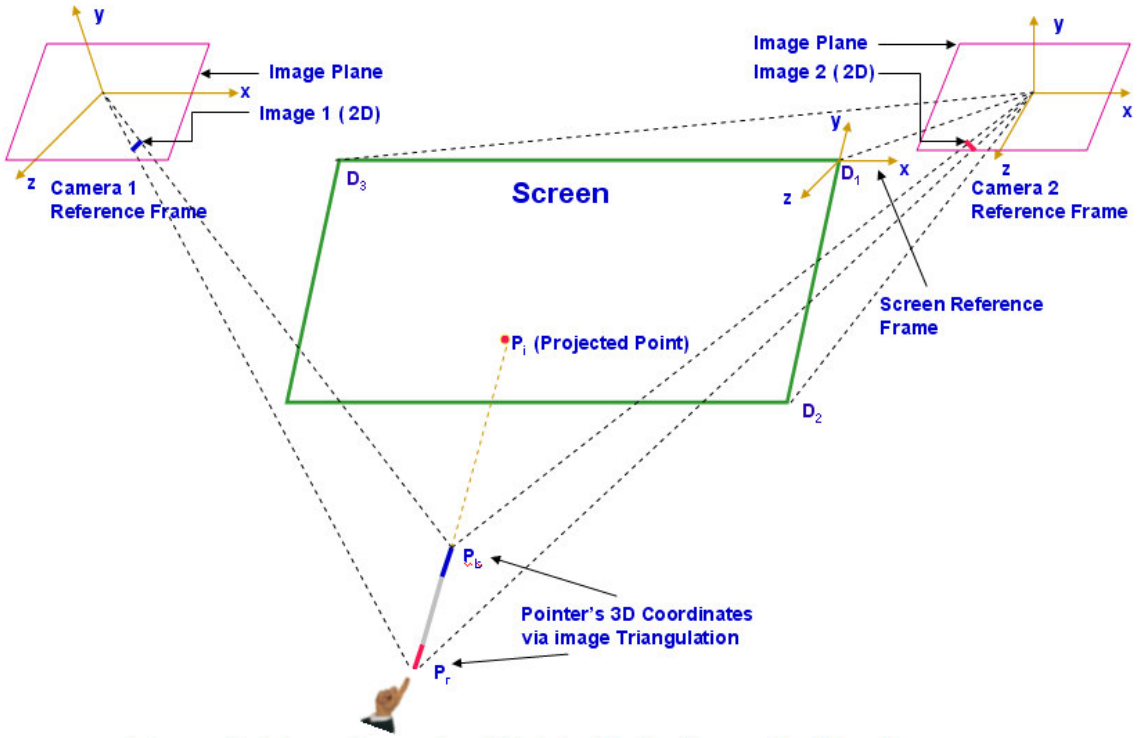

Figure 10: Three-dimensional Pointer Projection on the View Screen

2.5.1 Equation of a Plane Describing the View Screen

The standard equation of a plane in a 3D space is:

$$Ax + By + Cz + D = 0 \quad (23)$$

Where, the normal to the plane is the vector (A,B,C). Let us define the plane representing view screen by three points D_1 (x1,y1,z1), D_2 (x2,y2,z2), and D_3 (x3,y3,z3) in the camera 2 coordinate system (consistence with earlier calculations). The coefficients in Equation (23) are thus given by the following determinants.

$$A = \begin{vmatrix} 1 & y1 & z1 \\ 1 & y2 & z2 \\ 1 & y3 & z3 \end{vmatrix} \quad B = \begin{vmatrix} x1 & 1 & z1 \\ x2 & 1 & z2 \\ x3 & 1 & z3 \end{vmatrix} \quad C = \begin{vmatrix} x1 & y1 & 1 \\ x2 & y2 & 1 \\ x3 & y3 & 1 \end{vmatrix} \quad D = - \begin{vmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{vmatrix} \quad (24)$$

Further simplification to Equation (24) leads to

$$A = y1\,(z2 - z3) + y2\,(z3 - z1) + y3\,(z1 - z2) \quad (25)$$
$$B = z1\,(x2 - x3) + z2\,(x3 - x1) + z3\,(x1 - x2)$$
$$C = x1\,(y2 - y3) + x2\,(y3 - y1) + x3\,(y1 - y2)$$
$$D = -[x1\,(y2\,z3 - y3\,z2) + x2\,(y3\,z1 - y1\,z3) + x3\,(y1\,z2 - y2\,z1)]$$

Note that if the points are colinear then the normal (A,B,C) will be (0,0,0). The sign of s (which equals Ax + By + Cz + D) determines which side the point (x,y,z) lies with respect to the plane: if

s > 0 then the point lies on the same side as the normal (A,B,C); if s < 0 then it lies on the opposite side; if s = 0 then the point (x,y,z) lies on the plane.

2.5.2 Intersection of a Line and a Plane

The parametric representation of the equation of the line passing through points **Pr** (rx, ry, rz) and **Pb** (bx, by, bz) of the pointer is made as

$$\mathbf{P} = \mathbf{Pr} + u\,(\mathbf{Pb} - \mathbf{Pr}) \tag{26}$$

The point of intersection of the line and plane can be found by solving the system of equations represented by Eqs. (23) and (26). That is

$$A\,[rx + u\,(bx - rx)] + B\,[ry + u\,(by - ry)] + C\,[rz + u\,(bz - rz)] + D = 0 \tag{27}$$

Hence value of u is

$$u = \frac{A * rx + B * ry + C * rz + D}{A\,(rx - bx) + B(ry - by) + C(rz - bz)} \tag{28}$$

Substituting u into the equation of a line given by Eq (26) results in the point of intersection between line and plane as $\mathbf{P_i}$ shown in Figure 10. This projected point is where the pointer is pointing towards the view screen. Remember, the denominator of u in Eq. (26) is 0, the normal to the plane is perpendicular to the line. Thus the line is either parallel to the plane and there are no solutions or the line is on the plane in which case there are infinite solutions.

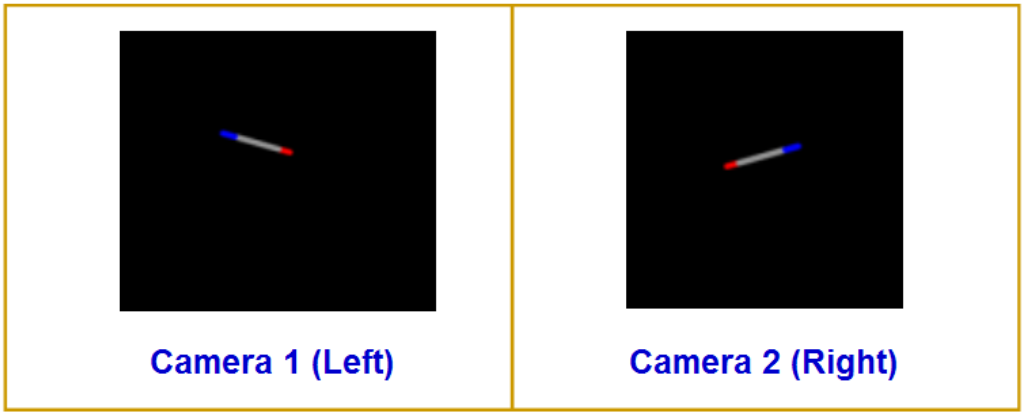

Figure 11: High Resolution Clipped Images (1920 x 1200 Pixels)

3. SIMULATION

A virtual pointer of size 2.7 inches was modeled with red and blue color ends in a large presentation room setting using the popular three-dimensional computer graphics and animation program called Studio 3D Max by Discreet, a subsidiary of Autodesk Inc. The pointer was situated at a distance of around 30 feet away from the view/presentation screen of size 12 feet x 3.5 feet. While the pointer was pointing towards view screen, two snap shots with high resolution (1920 x 1200 pixels) were taken form two cameras located near upper corner of the screen. Figure 11 depicts left- and right- camera static images clipped to reduce image size for presentation purpose.

Several configurations were simulated with different locations of the pointer in the room as well as various sizes of the pointer (small pointer: 2.7 in., medium pointer: 7 in. and large pointer: 21 in. in length). Furthermore, two pointers pointing towards screen as shown in Figure 12 were also investigated. Note that yellow and green colors of the second pointer were chosen primarily for purpose of image clarity.

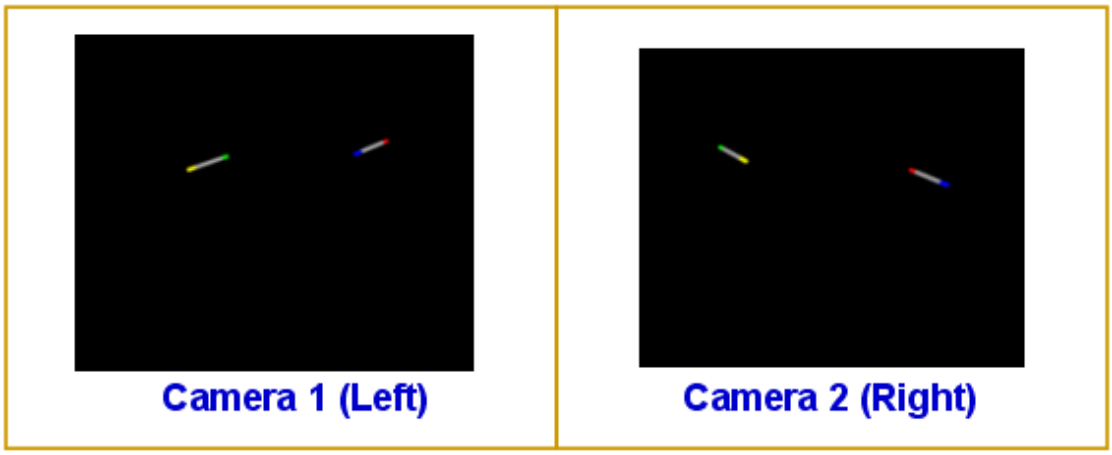

Figure 12: Double Pointer Tracking (1920 x 1200 Pixels)

Based on the modeling theory described in Section 2 (Methodology), a Visual C^{++} program was written to test the proposed analysis. The program takes images of two cameras as an input and computes the three-dimensional coordinates of the pointer (both position and orientation). In addition, the pointer's pointing projection on the view/presentation screen is outputted. These computed results were compared with the actual projections retrieved from Studio 3D Max.

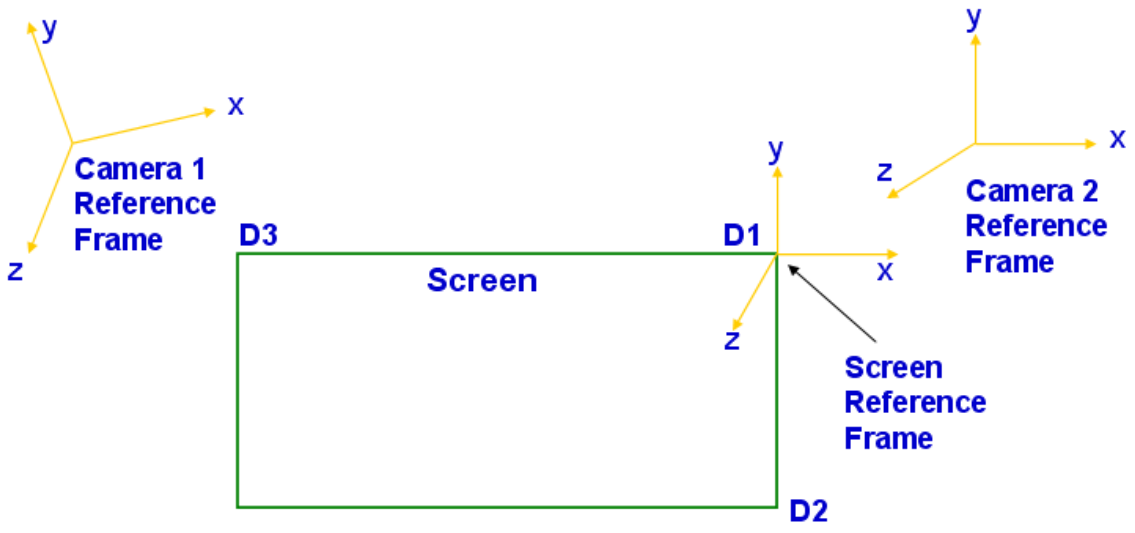

Figure 13: Definition of Reference Frames

4. RESULTS

Figure 13 describes various reference frames used for the analysis. The output results of the C++ program executions are grouped into three categories: one, the pointer's pointing accuracy on the view screen without rotating any cameras; two, when camera rotations are included in the analysis; and three, when variation in the pointer's sizes is included in the analysis. Table 1 presents five different test cases for the group one. The highlighted pink area describes changes in the configuration with respect to the case # 1. The output of the algorithm (the pointer's projection on the view screen) using triangulation method is compared to the corresponding retrieved values from the 3D Studio Max animation program. The worse case scenario is if it is of by 0.041 feet in y coordinate. The absolute average position accuracy for all five cases is 0.006 and 0.034 feet in the x and y coordinates, respectively. Note that the pointer is in neighborhood of 30 feet away from the screen. When the maximum absolute average accuracy (0.034 feet) is compared with 30 feet distance away from the screen, it is off by 0.11 % which is very small. Alternatively, compared to the size of the view screen (12 feet), it is off by around 0.28 %. However, it should be emphasized that the pointer's projection accuracy on the view screen does not depend upon what size of the screen is chosen in the analysis. Rather, it merely gives relative judgment on the pointing direction.

Test Case	Camera 1 Position			Camera 2 Position			Screen Position									Actual Screen Projection - Studio 3DMax		Computed Sreen Projection		Difference in Position	
							Point D1			Point D2			Point D3								
	x	y	z	x	y	z	x	y	z	x	y	z	x	y	z	x	y	x	y	x	y
	pitch	yaw	roll	pitch	yaw	roll															
1	-12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	1.7	-5.999	1.663	-0.001	0.037
	0.0	0.0	0.0	0.0	0.0	0.0															
2	-12.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	1.7	-5.983	1.738	-0.017	-0.038
Camera shift	0.0	0.0	0.0	0.0	0.0	0.0															
3	-12.0	0.0	0.0	0.0	0.0	0.0	-0.37	2.745	0.583	-0.45	3.939	-2.71	-11.5	-1.34	-0.61	-6.0	1.7	-6	1.659	0.000	0.041
Screen rotate	0.0	0.0	0.0	0.0	0.0	0.0															
4	-12.0	0.0	0.0	0.0	0.0	0.0															
Pointer move	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	-1.7	-5.999	-1.663	-0.001	-0.037
5	-12.0	0.0	0.0	0.0	0.0	0.0															
Pointer move	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-5.0	-3.0	-5.012	-2.984	0.012	-0.016
	All dimensions are in feet. Highlighted area describes the changes with respect to case # 1																Absolute Average			0.006	0.034

Table 1: Positioning Accuracy Comarison

Table 2 presents the results when camera rotations are included. Note that case # 9 accounts for all three-axis camera rotations. These specific angles are considered in order to maximize view coverage of the presentation room. The accuracy in this category is relatively poor due to errors in rotational calibration of the camera. This can be considerably improved upon choosing appropriate calibration techniques.

Test Case	Camera 1 Position			Camera 2 Position			Screen Position									Actual Screen Projection - Studio 3DMax		Computed Screen Projection		Difference in Position	
							Point D1			Point D2			Point D3								
	x	y	z	x	y	z	x	y	z	x	y	z	x	y	z	x	y	x	y	x	y
	pitch	yaw	roll	pitch	yaw	roll															
6	-12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	1.7	-6.677	1.484	0.677	0.216
Yaw rotation	0.0	16.31	0.0	0.0	0.0	0.0															
7	-12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	1.7	-5.777	1.484	-0.223	0.216
Pitch rotation	11.31	0.0	0.0	0.0	0.0	0.0															
8	-12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	1.7	-6.002	1.755	0.002	-0.055
Roll roation	0.00	0.0	24.0	0.0	0.0	0.0															
9	-12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	-1.7	-6.747	-1.718	0.747	0.018
Pitch-Yaw-Roll	10.00	12.0	14.0	0.0	0.0	0.0															
																	Absolute Average			0.412	0.126

All Dimensions are in feet. Pitch, Yaw and Roll are in degrees.
Highlighted area describes the changes with respect to case # 1

Table 2: Rotational Accuracy Comarison

Table 3 considers the variation in the pointer's length. It is very encouraging to see that the smallest pointer of size 2.7 inches was detected with a high accuracy even with both cameras rotated. Also, the size of the pointer does not have much effect in the analysis.

Test Case	Camera 1 Position			Camera 2 Position			Screen Position									Actual Screen Projection - Studio 3DMax		Computed Screen Projection		Difference in Position	
							Point D1			Point D2			Point D3								
	x	y	z	x	y	z	x	y	z	x	y	z	x	y	z	x	y	x	y	x	y
	pitch	yaw	roll	pitch	yaw	roll															
10	-12.0	0.0	0.0	0.0	0.0	0.0															
Small pointer	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	-1.7	-6.001	-1.8	0.001	0.100
11	-12.0	0.0	0.0	0.0	0.0	0.0															
Medium pointer	3.687	6.896	-8.01	-12.1	-6.21	-4.00	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	-1.7	-6.024	-1.642	0.024	-0.058
12	-12.0	0.0	0.0	0.0	0.0	0.0															
Large pointer	0.0	0.0	0.0	0.0	-51.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	-1.7	-5.727	-1.779	-0.273	0.079
																	Absolute Average			0.099	0.047

All Dimensions are in feet. Pitch, Yaw and Roll are in degrees.
Highlighted area describes the changes with respect to case # 1
Small pointer size = 2.7 in., Medium pointer size = 7 in., and Large pointer size = 21 in.

Table 3: Length Accuracy Comarison

5. CONCLUSION

The analysis reveals that the image triangulation method works reasonably well for locating the pointer in a relatively large three-dimensional room space. Furthermore, the pointer's projections on the view screens are accurate well within many presenter-audience applications. The computational errors are considered to be small when one view the screen from the audience located in the neighborhood of 30 feet away where precise visualization of pointer's direction is not that clear. Future investigation includes choosing actual hardware in the loop and incorporating most recent image enhancement/ detection schemes [Refs. 11 - 13].

6. ACKNOWLEGEMENT

The authors wish to acknowledge AFRL, Information Directorate, US Air Force, Rome, New York for their continuing support for this research work.

7. REFERENCES

1. DataWall information: www.if.afrl.af.mil/tech/programs/ADII/adii_main.html

2. B. Moghaddam and A. Pentland, *"Maximum Likelihood of Detection of Faces and Hands",* Proceeding of International Workshop on Automatic Face and Gesture Recognition, 122 -128, 1995

3. P. Cipolla and N. Hollinghurst, *"Uncalibrated Stereo Vision with Pointing for a Man-Machine Interface"*, Proceedings of IAPR Workshop on Machine Vision Applications, 163 – 166, 1994

4. L. Gupta and S. Ma, *"Gesture-Based Interaction and Communication: Automated Classification of Hand Gesture Contours",* IEEE Transactions on System, Man an Cybernetics – Part C: Applications and Reviews, 31 (1), 114 - 120, Feb 2001

5. R. Tsai, *"An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision",* Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL., 363 – 374, 1986

6. O. Faugeras and G. Toscani, *"The Calibration Problem for Stereo",* International Proceedings of CVPR, 15 – 20, 1986

7. B. Caprile and V. Torre, *"Using Vanishing Points for Camera Calibration",* International Journal of Computer Vision, Vol. 3, 127 – 140, 1990

8. S. Maybank and O. D. Faugeras, *"A Theory of Self-Calibration of a Moving Camera",* International Journal of Computer Vision, Vol. 8, 123 – 151, 1992

9. D. Tzovaras, N. Grammalidis and M. G. Stromtzis, *" Object-based Coding of Stereo Image Sequences using Joint 3D Motion / Disparity Compensation",* IEEE Transaction on Circuits System Video Tech., Vol. 7, 312 – 327, April 1997

10. R. Jain, R. Kasturi and B. Schunck, *Machine Vision*, McGraw Hill, Inc. (1995)

11. R. Maini and H. Aggrawal, *"Study and Comparison of Various Image Edge Detection Techniques",* International Journal of Image Processing, Computer Science Journals, 3 (1), 1 – 12, January/ February 2009

12. A. Mohammed and S. Rusthum, *"Object-Oriented Image Processing of an High Resolution Satellite Imagery with Perspectives for Urban Growth, Planning and Development",* International Journal of Image Processing, Computer Science Journals, 2 (3), 18 -28, May/ June 2008

13. P. Hiremath and J. Pujari, *"Content Based Image Retrieval using Color Boosted Salient Points and Shape Features of an Image"*, International Journal of Image Processing, Computer Science Journals, 2 (1), 10 – 17, January/ February 2008