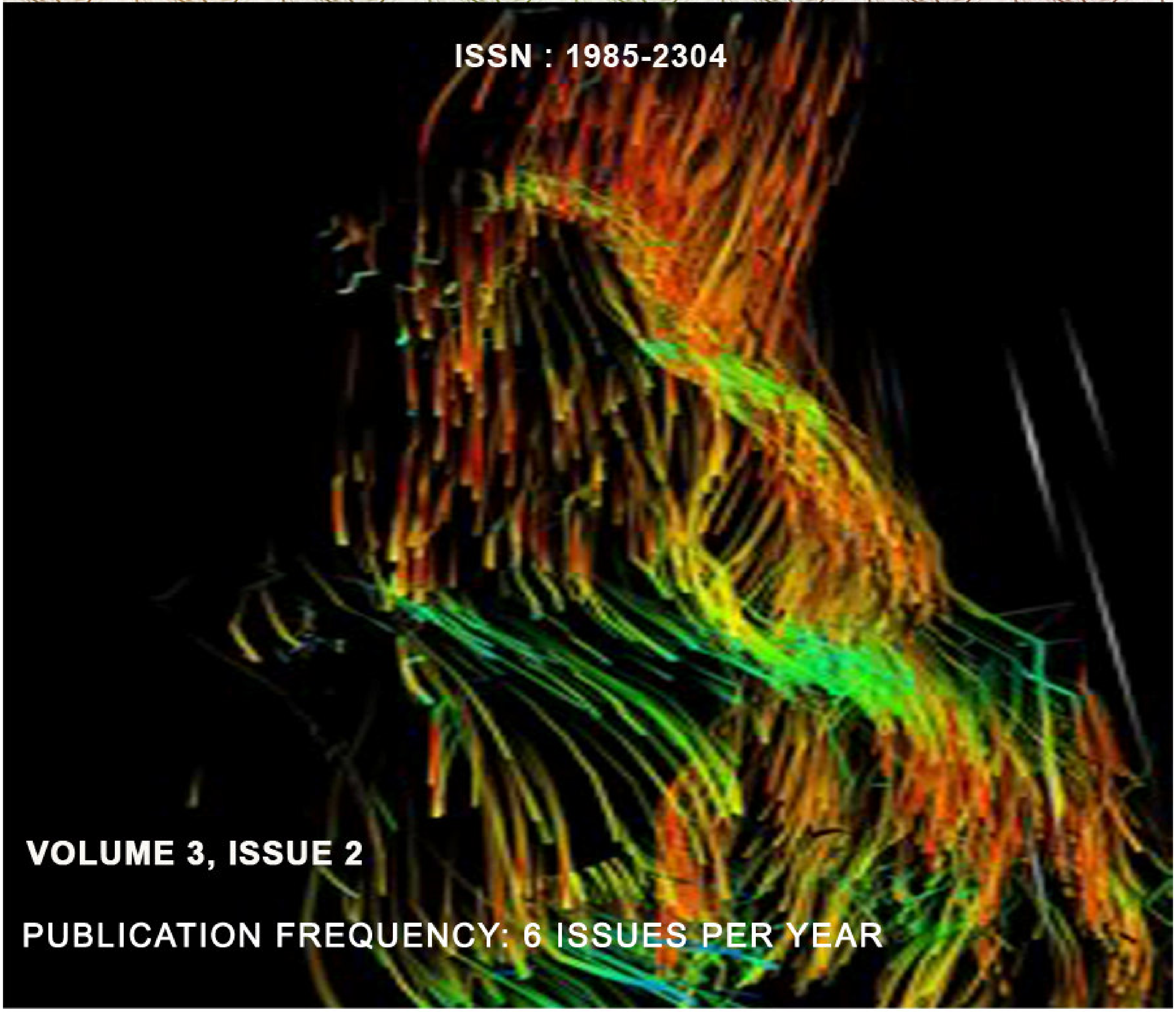
International Journal of Image Processing (IJIP)



ISSN : 1985-2304

VOLUME 3, ISSUE 2

PUBLICATION FREQUENCY: 6 ISSUES PER YEAR

Editor in Chief Professor Hu, Yu-Chen

International Journal of Image Processing (IJIP)

Book: 2009 Volume 3, Issue 2
Publishing Date: 31-04-2009
Proceedings
ISSN (Online): 1985-2304

IJIP Journal is a part of CSC Publishers
http://www.cscjournals.org

©IJIP Journal
Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers

Table of Contents

Volume 3, Issue 2, April 2009.

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.

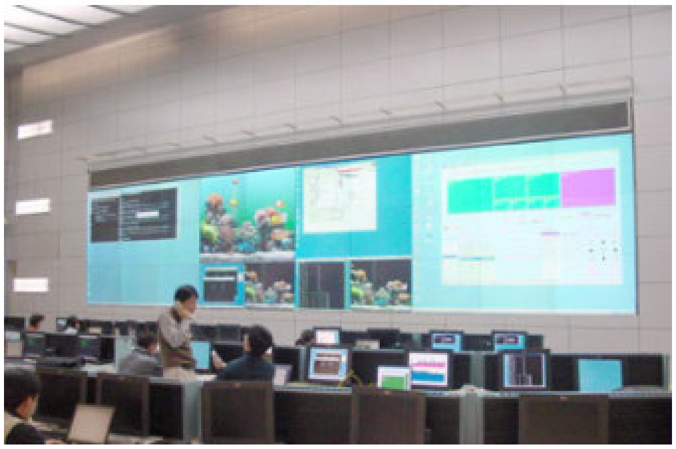


Figure 1c: Commercial Bank

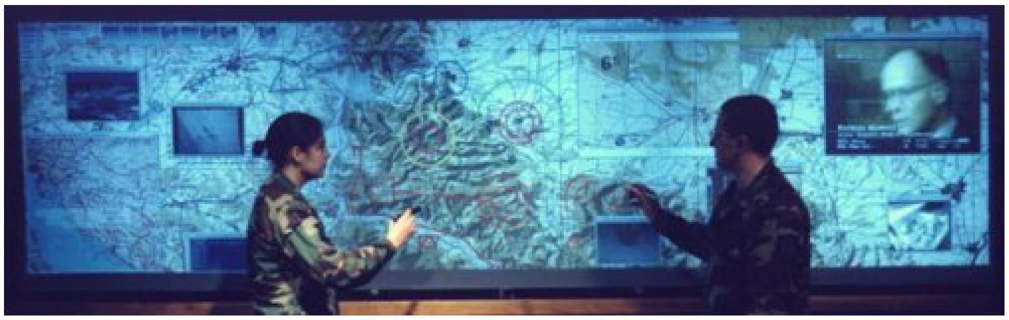


Figure 1d: Interactive DataWall of AFRL, US Air Force

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.

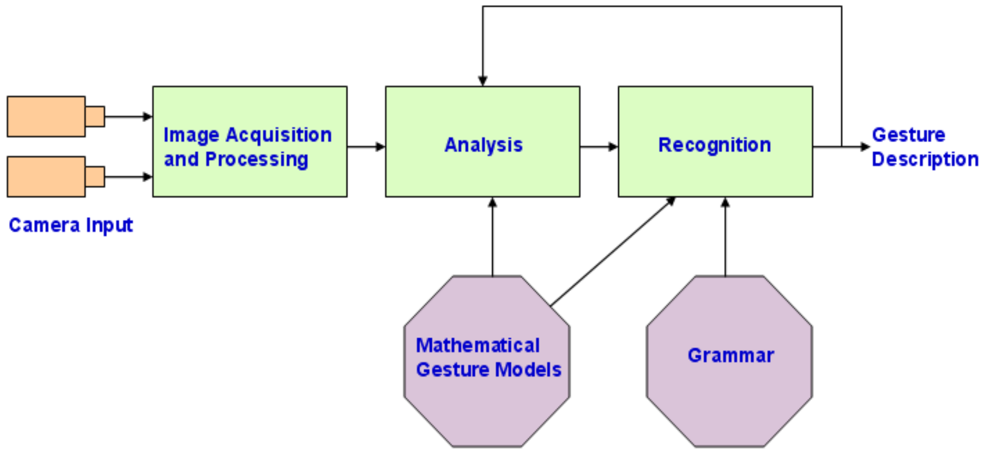


Figure 2: Gesture Interpretation System

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.

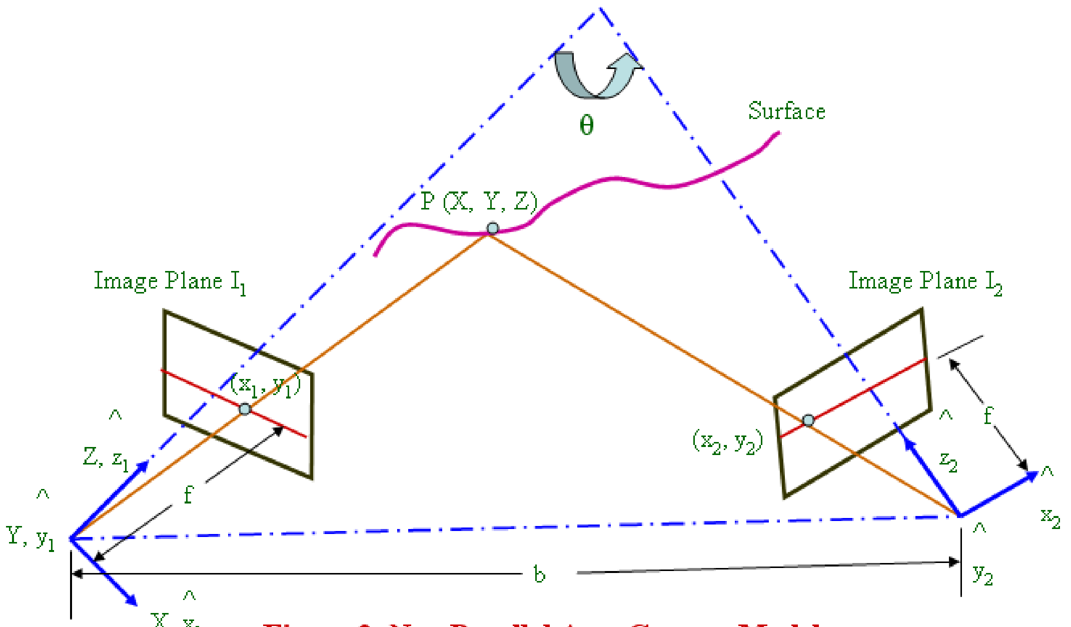


Figure 3: Non Parallel Axes Camera Model

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.

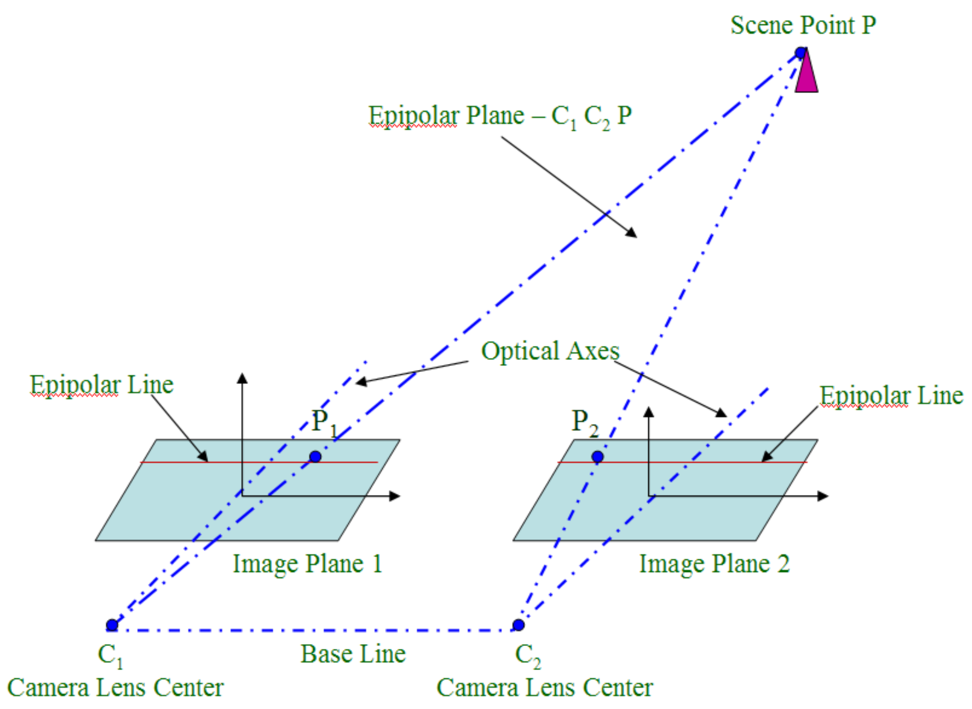


Figure 4: The Epipolar Plane

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

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

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

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

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 **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)

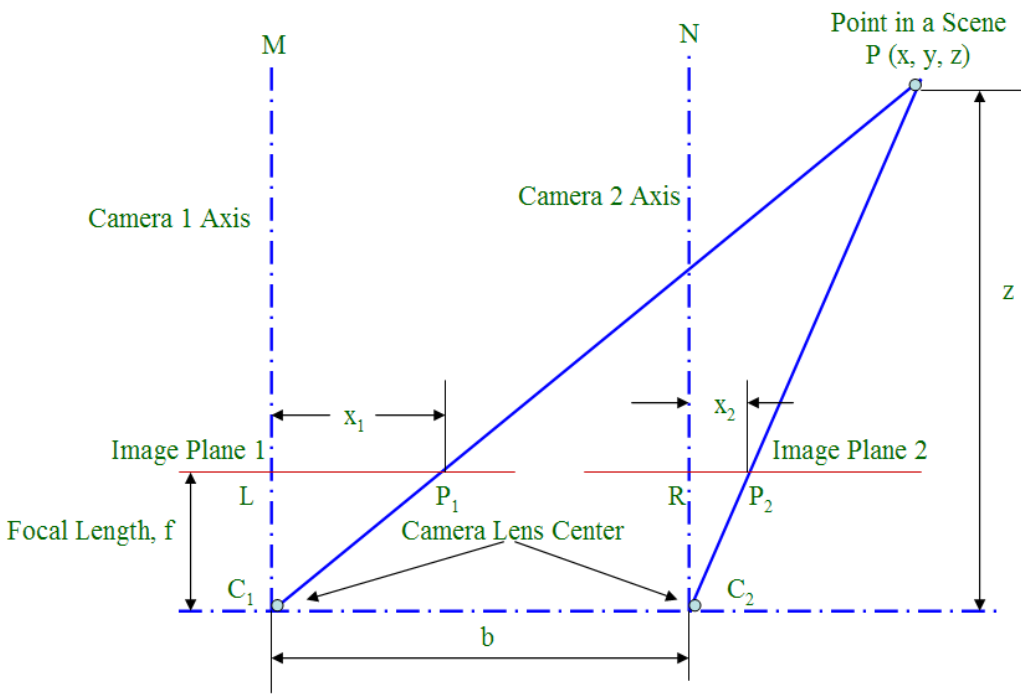


Figure 5: The Parallel Axes Camera Model

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

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

$$\mathbf{P}(I, J) = R(I, J)\,\mathbf{e}_1 + G(I, J)\,\mathbf{e}_2 + B(I, J)\,\mathbf{e}_3 \qquad (9)$$

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

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

for the red color end pixels and

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

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

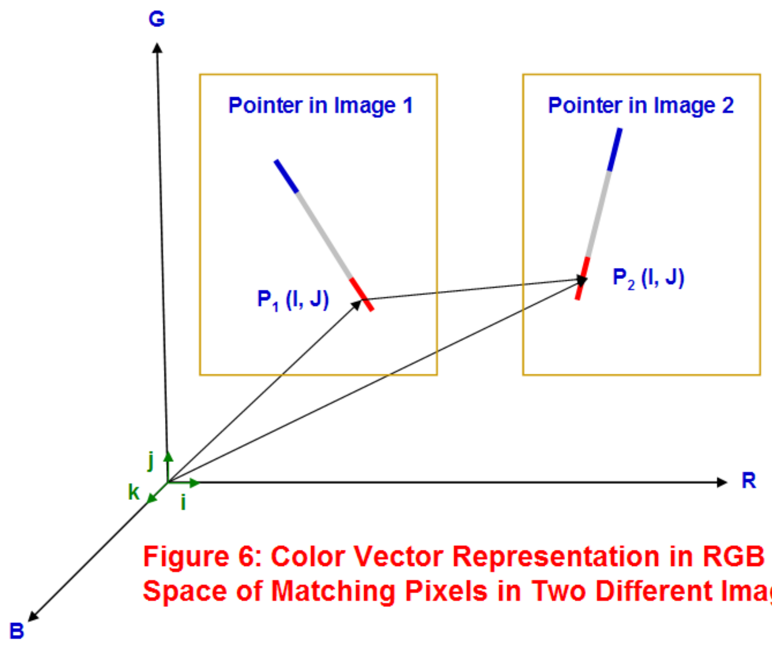


Figure 6: Color Vector Representation in RGB Space of Matching Pixels in Two Different Images

$$\mathbf{P}_1(I, J) = R_1(Imid, Jmid) \qquad (12)$$

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

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

We assume that the centroid points **P₁** (I, J) and **P₂** (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).

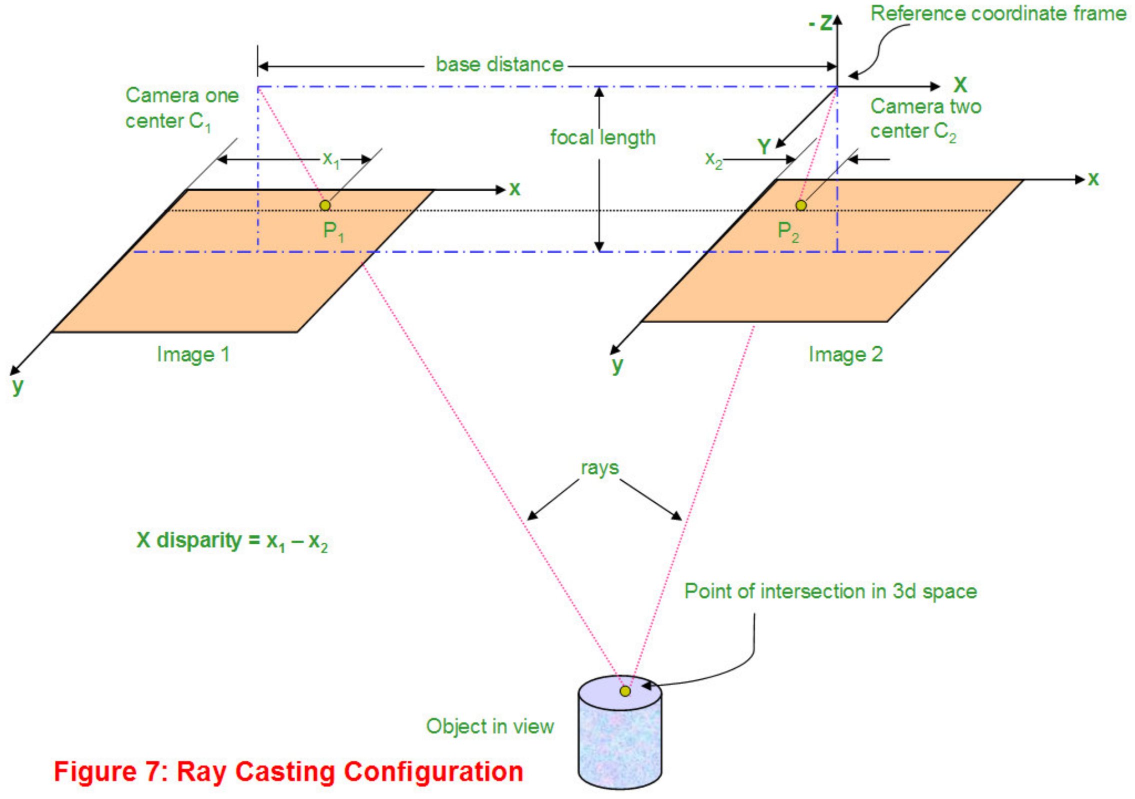


Figure 7: Ray Casting Configuration

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.

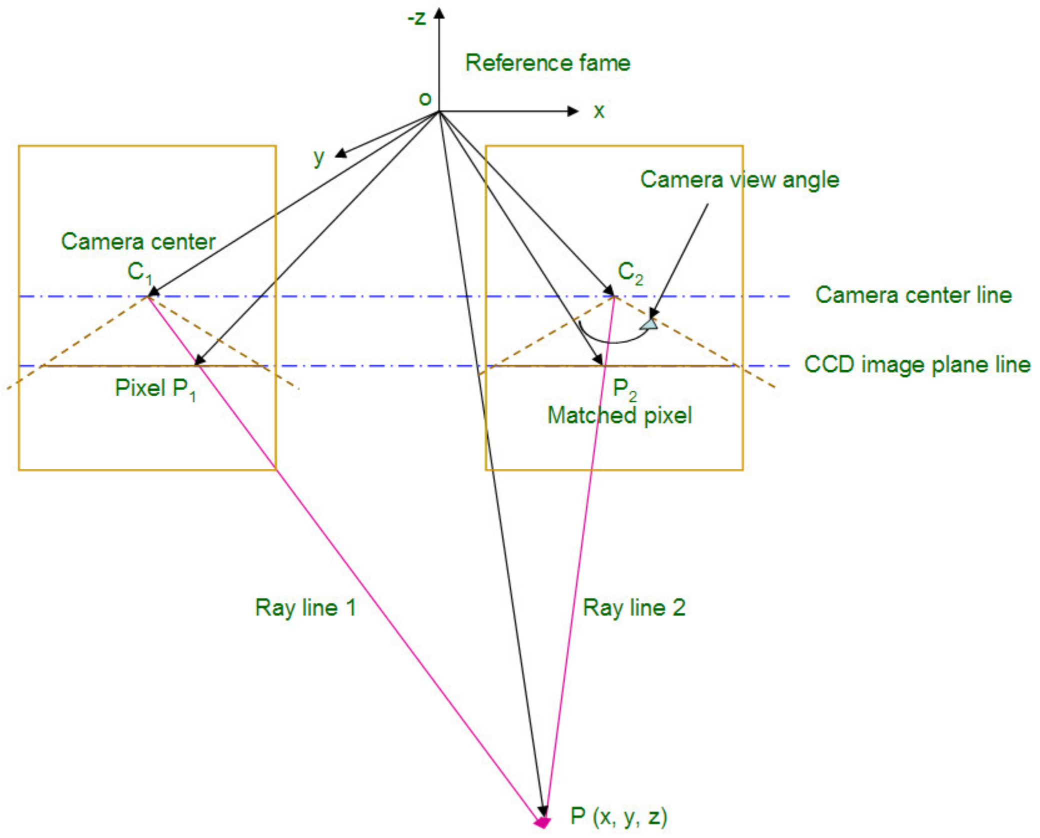


Figure 8: Coordinate Computation for Two Lines of Intersection

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:

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

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

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

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

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

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

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

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

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

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

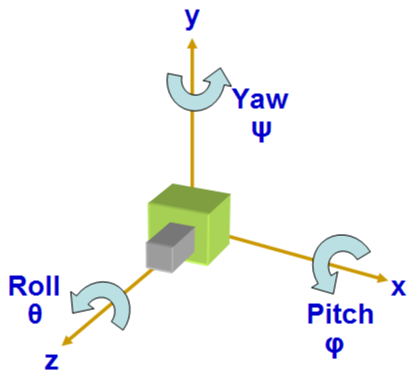


Figure 9: Camera's Pitch Yaw and Roll Axes

$$R\ (x,\ \text{pitch}) = R\ (x,\ \varphi) = \left\{\begin{matrix} 1 & 0 & 0 \\ 0 & C\varphi & -S\varphi \\ 0 & S\varphi & C\varphi \end{matrix}\right\} \tag{18}$$

$$R\ (y,\ \text{yaw}) = R\ (y,\ \psi) = \left\{\begin{matrix} C\psi & 0 & S\psi \\ 0 & 1 & 0 \\ -S\psi & 0 & C\psi \end{matrix}\right\} \tag{19}$$

$$R\ (z,\ \text{roll}) = R\ (z,\ \theta\ ) = \left\{\begin{matrix} C\theta & -S\theta & 0 \\ S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{matrix}\right\} \tag{20}$$

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

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

$$= \left\{\begin{matrix} C\theta\ C\psi & -\ S\theta\ C\psi & S\psi \\ C\theta\ S\varphi S\psi + C\varphi\ S\theta & C\theta\ C\varphi\ -\ S\theta\ S\varphi\ S\psi & -\ C\psi\ S\varphi \\ S\theta\ S\varphi\ -\ C\theta\ C\varphi\ S\psi & S\theta C\varphi\ S\psi + C\theta S\varphi & C\varphi\ C\psi \end{matrix}\right\}$$

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

$$\left\{\begin{matrix} x \\ y \\ z \end{matrix}\right\} = \left\{PYR\right\} \left\{\begin{matrix} x' \\ y' \\ z' \end{matrix}\right\} \qquad \left\{\begin{matrix} x' \\ y' \\ z' \end{matrix}\right\} = \left\{PYR\right\}^{-1} \left\{\begin{matrix} x \\ y \\ z \end{matrix}\right\} \tag{22}$$

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.

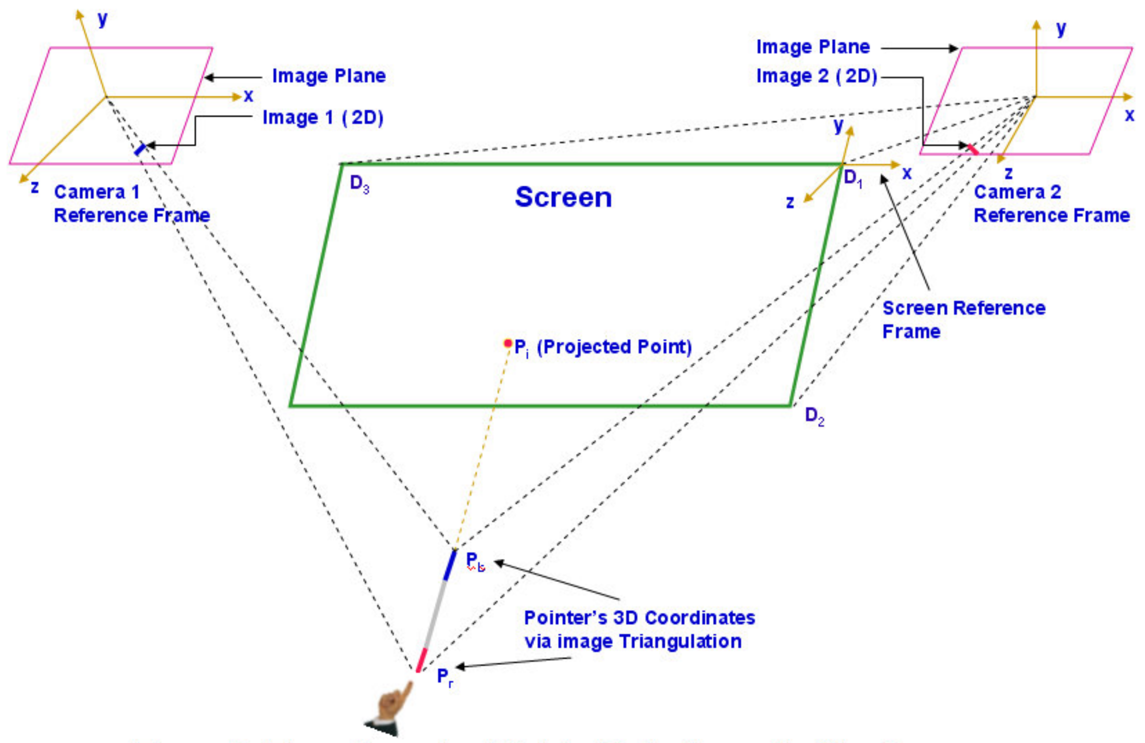


Figure 10: Three-dimensional Pointer Projection on the View Screen

2.5.1 Equation of a Plane Describing the View Screen

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

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

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.

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

Further simplification to Equation (24) leads to

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

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

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

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

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

Hence value of u is

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

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.

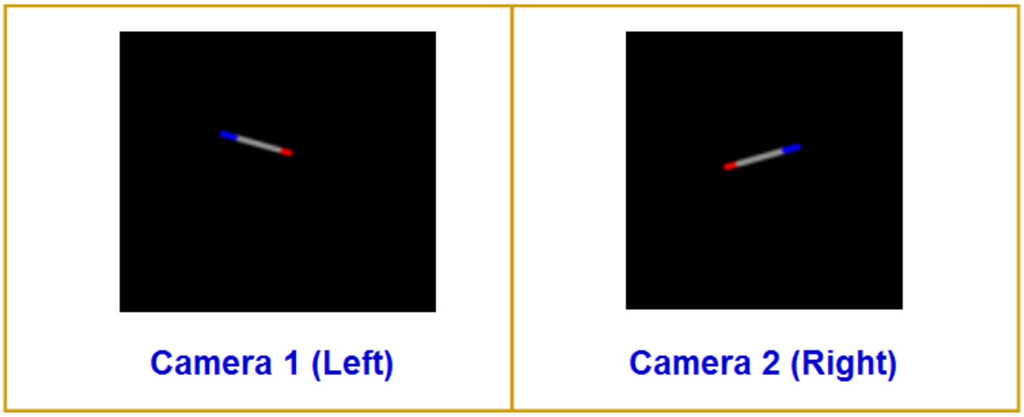


Figure 11: High Resolution Clipped Images (1920 x 1200 Pixels)

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.

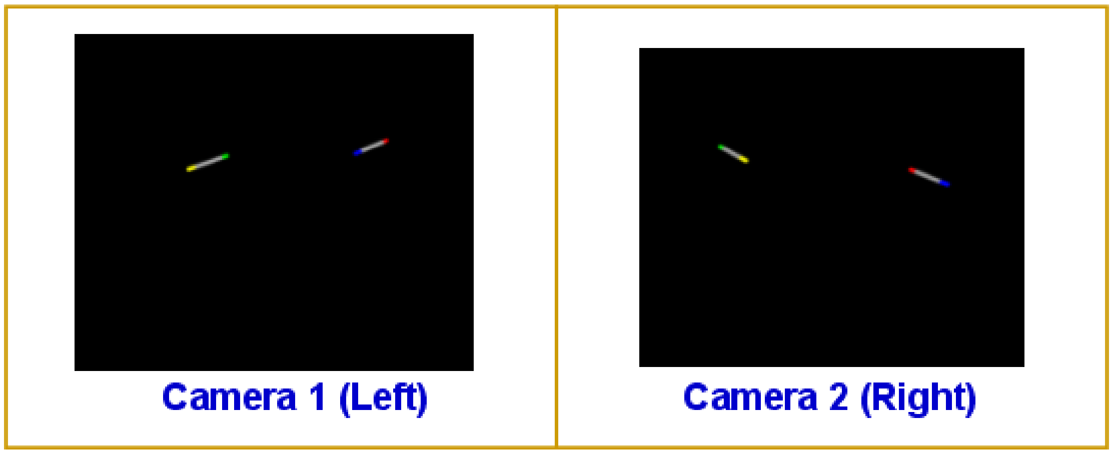


Figure 12: Double Pointer Tracking (1920 x 1200 Pixels)

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.

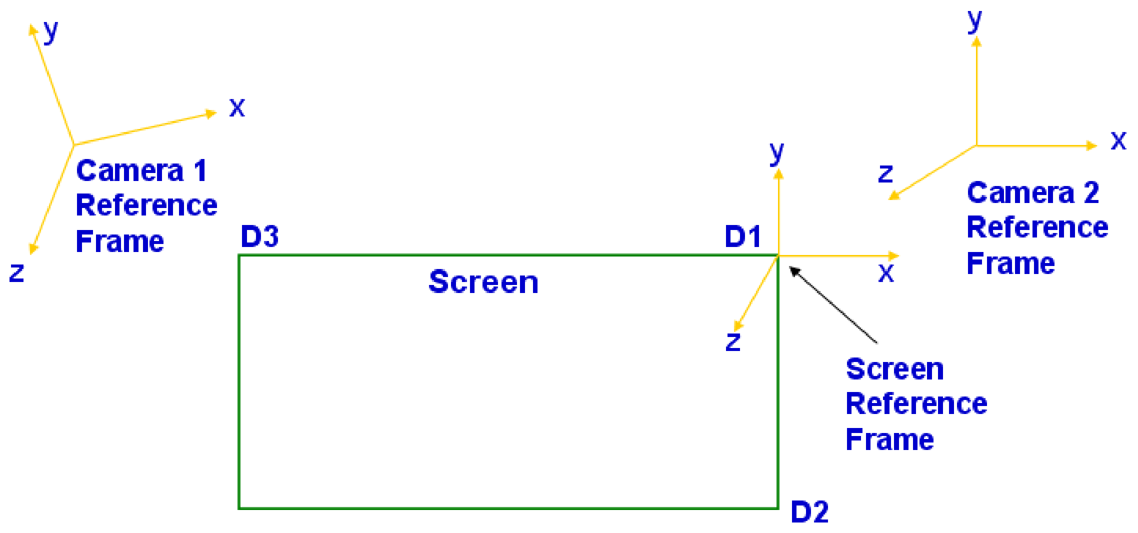
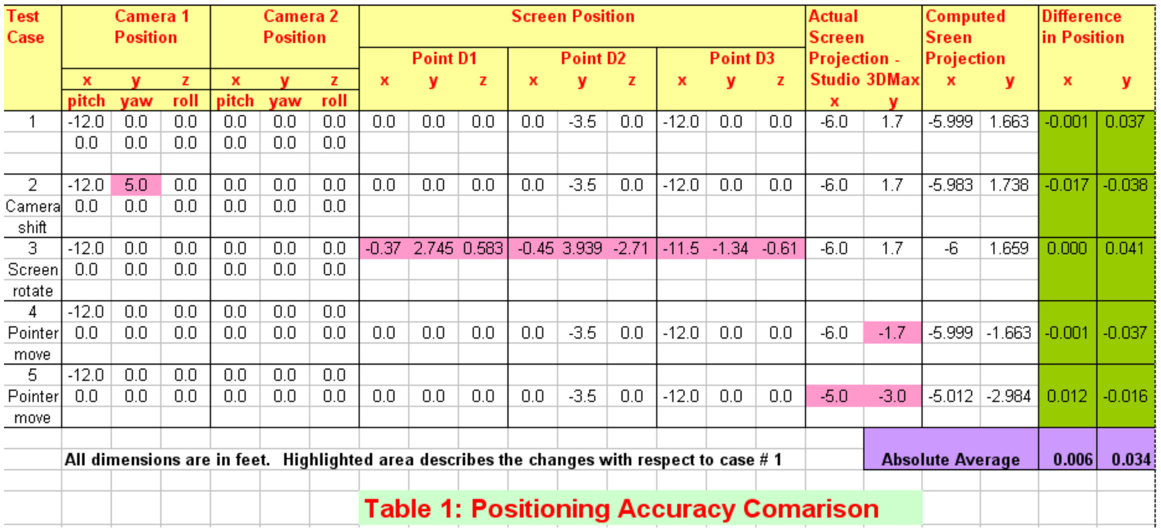


Figure 13: Definition of Reference Frames

4. RESULTS

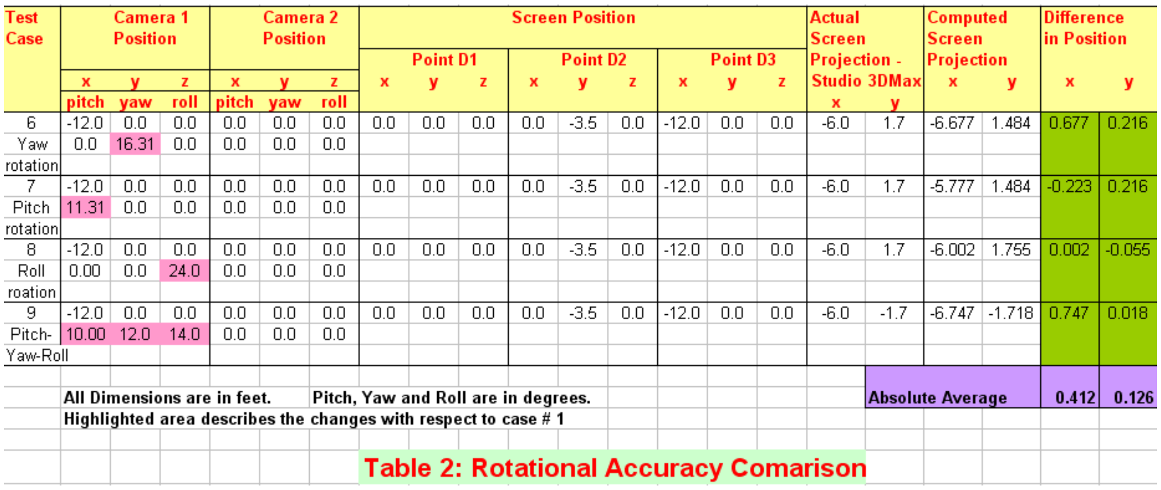
Figure 13 describes various reference frames used for the analysis. The output results of the C^{++} program executions are grouped into three categories: one, the pointer's pointing accuracy on the view screen without rotating any cameras; two, when camera rotations are included in the analysis; and 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
	pitch	yaw	roll	pitch	yaw	roll										x	y				
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	0.0	0.0	0.0	0.0	0.0	0.0															
shift																					
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	0.0	0.0	0.0	0.0	0.0	0.0															
rotate																					
4	-12.0	0.0	0.0	0.0	0.0	0.0															
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	-5.999	-1.663	-0.001	-0.037
move																					
5	-12.0	0.0	0.0	0.0	0.0	0.0															
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	-5.0	-3.0	-5.012	-2.984	0.012	-0.016
move																					
	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.

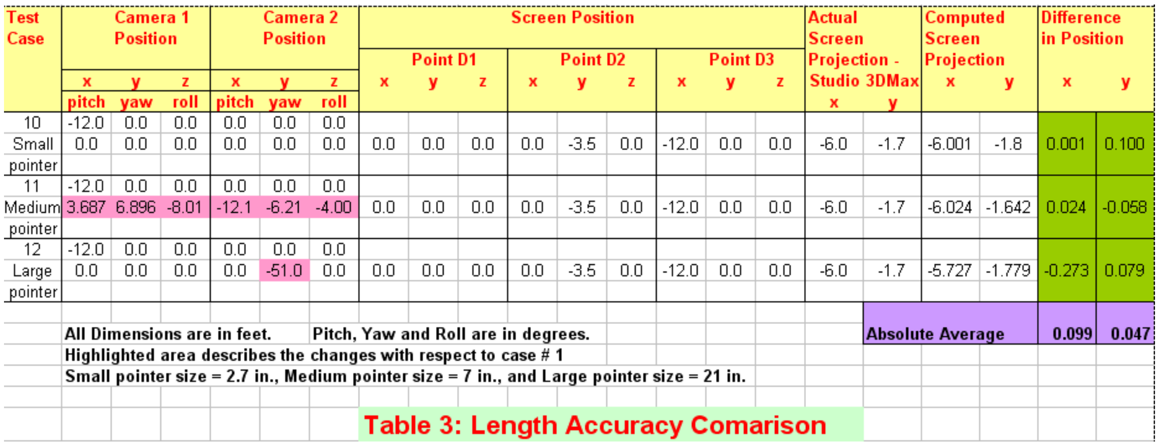


Test Case	Camera 1 Position			Camera 2 Position			Screen Position									Actual Screen Projection - Studio 3DMax		Computed Screen Projection		Difference in Position	
							Point D1			Point D2			Point D3								
	x	y	z	x	y	z	x	y	z	x	y	z	x	y	z	x	y	x	y	x	y
	pitch	yaw	roll	pitch	yaw	roll															
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	0.0	16.31	0.0	0.0	0.0	0.0															
rotation																					
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	11.31	0.0	0.0	0.0	0.0	0.0															
rotation																					
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	0.00	0.0	24.0	0.0	0.0	0.0															
roation																					
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-	10.00	12.0	14.0	0.0	0.0	0.0															
Yaw-Roll																					
																	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	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
pointer																					
11	-12.0	0.0	0.0	0.0	0.0	0.0															
Medium	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
pointer																					
12	-12.0	0.0	0.0	0.0	0.0	0.0															
Large	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
pointer																					
																	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

A Framework for Human Action Detection via Extraction of Multimodal Features

Lili N. A. liyana@fsktm.upm.edu.my
Department of Multimedia
Faculty of Computer Science & Information Technology
University Putra Malaysia
43400 UPM Serdang
Selangor, Malaysia

Abstract

This work discusses the application of an Artificial Intelligence technique called data extraction and a process-based ontology in constructing experimental qualitative models for video retrieval and detection. We present a framework architecture that uses multimodality features as the knowledge representation scheme to model the behaviors of a number of human actions in the video scenes. The main focus of this paper placed on the design of two main components (model classifier and inference engine) for a tool abbreviated as VASD (Video Action Scene Detector) for retrieving and detecting human actions from video scenes. The discussion starts by presenting the workflow of the retrieving and detection process and the automated model classifier construction logic. We then move on to demonstrate how the constructed classifiers can be used with multimodality features for detecting human actions. Finally, behavioral explanation manifestation is discussed. The simulator is implemented in bilingual; Math Lab and C++ are at the backend supplying data and theories while Java handles all front-end GUI and action pattern updating. To compare the usefulness of the proposed framework, several experiments were conducted and the results were obtained by using visual features only (77.89% for precision; 72.10% for recall), audio features only (62.52% for precision; 48.93% for recall) and combined audiovisual (90.35% for precision; 90.65% for recall).

Keywords: audiovisual, human action detection, multimodal, hidden markov model.

1. INTRODUCTION

Action is the key content of all other contents in the video. Action recognition is a new technology with many potential applications. Action recognition can be described as the analysis and recognition of human motion patterns. Understanding activities of objects, especially humans, moving in a scene by the use of video is both a challenging scientific problem and a very fertile domain with many promising applications. Use of both audio and visual information to recognize actions of human present might help to extract information that would improve the recognition results. What to argue is that action is the key content of all other contents in the video. Just imagine describing video content effectively without using a verb. A verb is just a description (or expression) of actions. Action recognition will provide new methods to generate video retrieval and categorization in terms of high-level semantics.

When either audio or visual information alone is not sufficient, combining audio and visual features may resolve the ambiguities and to help to obtain more accurate answers Unlike the traditional methods that analyze audio and video data separately, this research presents a method which able to integrate audio and visual information for action scene analysis. The approach is top-down for determining and extract action scenes in video by analyzing both audio and visual data. A multidimensional layer framework was proposed to detect action scene automatically. The first level extracts low level features such as motion, edge and colour to detect video shots and next we use Hidden Markov model(HMM) to detect the action. An audio feature vector consisting of *n* audio features which is computed over each short audio clip for the purpose of audio segmentation was used too. Then it is time to decide the fusion process according to the correspondences between the audio and the video scene boundaries using an HMM-based statistical approach. Results are provided which prove the validity of the approach. The approach consists of two stages: audiovisual event and semantic context detections. HMMs are used to model basic audio events, and event detection is performed. Then semantic context detection is achieved based on Gaussian mixture models, which model the correlations among several action events temporally. With this framework, the gaps between low-level features and the semantic contexts that last in a time series was bridged. The experimental evaluations indicate that the approach is effective in detecting high-level semantics such as action scene.

2. RELATED WORKS

The problem of human action recognition is complicated by the complexity and variability of shape and movement of the human body, which can be modelled as an articulated rigid body. Moreover, two actions can occur simultaneously, e.g., walk and wave. Most work on action recognition involving the full human body is concerned with actions completely described by motion of the human body, i.e., without considering interactions with objects.

Previous works on audio visual content analysis were quite limited and still at a preliminary stage. Most of the approaches are focused on visual information such as colour histogram differences, motion vectors and key frames [3,4,5]. Colour histogram [7] difference and motion vector between video frames or objects are the most common features in the scene recognition algorithms. Although such features are quite successful in the video shot segmentation, scene detection based on such visual features alone poses many problems. The extraction of relevant visual features from an images sequence, and interpretation of this information for the purpose of recognition and learning is a challenging task. An adequate choice of visual features is very important for the success of action recognition systems [6].

Because of the different sets of genre classes and the different collections of video clips these previous works chose, it is difficult to compare the performance of the different features and approaches they used.

Automatic interpretation of human action in videos has been actively done for the past years. The investigation of human motion has drawn a great attention from researches in computer vision or computer graphics recently. Fruitful results can be found in many applications such as visual surveillance. Features at different levels have been proposed for human activity analysis. Basic image features based on motion histogram of objects are simple and reliable to compute (Efros et al. 2003). In (Stauffer & Grimson 2000), a stable, real-time outdoor tracker is proposed, and high-level classifications are based on blobs and trajectories output from this tracking system. In (Zelnik-Manor & Irani 2001), dynamic actions are regarded as long-term temporal objects, and spatio-temporal features at multiple temporal. There is a huge body of literature on the topics of visual tracking, motion computation, and action detection. As tools and systems for producing and disseminating action data improve significantly, the amount of human action detection system grows rapidly. Therefore, an efficient approach to search and retrieve human action data is needed.

3. METHODOLOGY

The process of video action detection is depicted in Figure 1.0.

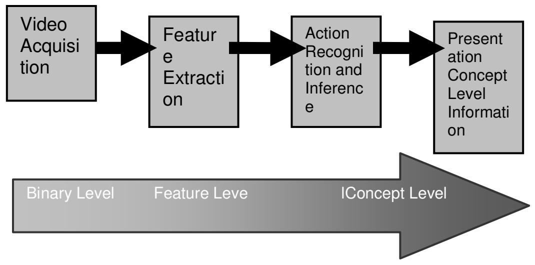


Figure 1.0 Metadata Generation Process

There are three classes of information. The main type of information, which serves as the input of the complete system, is the *binary level information, which* comprises the raw video files. This binary level information is analyzed which results in the so called *feature level information*, i.e. features interesting for detecting certain action like the measurements of action speeds. Finally, from this feature level information, actions are detected that are regarded as the *concept level information*. When a user searches information from the multimedia, it does not have to browse the binary level video files anymore, but it can directly query on concept level. The user can for example query an action in which a man is punching another man, which might cause a violent later on. As a result a more advanced search engine is created that can be used for human action purposes.

In this paper, the feature extraction process (Figure 2.0) is discussed: converting the binary level information coming from the video acquisition system into feature level information suited for further processing. From the flowchart (Figure 3.0), we have four levels of processes:

- Pixel level represents the average percent of the changed pixels between frames within a shot
- Histogram indicates the mean value of the histogram difference between frames within a shot
- Segmentation level to indicate background and foreground areas
- Object tracking based on observation: flames, explosion, gun

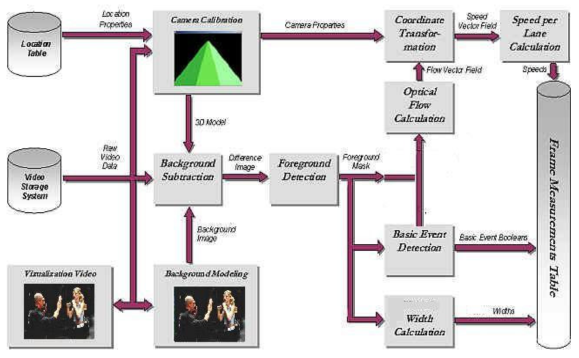


Figure 2.0 The overview of the system

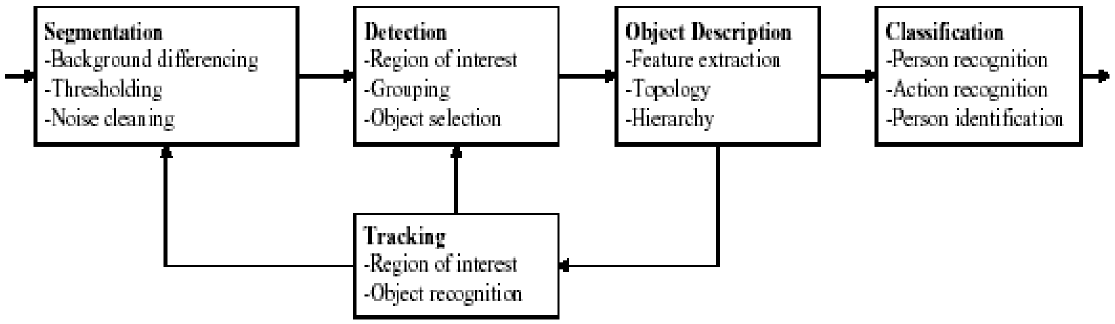


Figure 3.0 Flow chart of feature extraction process

We proposed an action classification method with the following characteristics:

- Feature extraction is done automatically;
- The method deals with both visual and auditory information, and captures both spatial and temporal characteristics;
- Edge feature extraction
- Motion
- Shot activity that conveys large amount of information about the type of video
- Colour feature extraction
- Sound
- The extracted features are natural, in the sense that they are closely related to the human perceptual processing.

3.1 Segmentation

The segmentation and detection stages are often combined and simply called object detection. Basically the task of the segmentation is to split the image into several regions based on color, motion or texture information, whereas the detection stage has to choose relevant regions and assign objects for further processing.

We follow the assumption that the video sequence is acquired using a stationary camera and there is only very little background clutter. Based on this we use background differencing followed by threshold to obtain a binary mask of the foreground region. In order to remove noise median filtering and morphological operations are used. Regions of interest (ROI) are detected using boundary extraction and a simple criterion based on the length of the boundary. Boundary filling is applied to each ROI and the resulting binary object masks are given to the description stage.

3.2 Object Descriptor

The object description refers to a set of features that describe the detected object in terms of color, shape, texture, motion, size etc. The goal of the feature extraction process is to reduce the existing information in the image into a manageable amount of relevant properties. This leads to a lower complexity and a more robust description. Additionally, the spatial arrangement of the objects within the video frame and as related to each other is characterized by a topology.

A very important issue for the performance of a subsequent classification task is to select a suitable descriptor that expresses both the similarity within a class and the distinctions between different classes. Since the classifier strongly depends on the information provided by the descriptor it is necessary to know its properties and limitations relating to this specific task. In case of human body posture recognition it is obvious that shape descriptors are needed to extract useful information.

3.3 Classification

Classification is a pattern recognition (PR) problem of assigning an object to a class. Thus the output of the PR system is an integer label. The task of the classifier is to partition the feature space into class-labeled decision regions. Basically, classifiers can be divided into parametric and non-parametric systems depending on whether they use statistical knowledge of the observation and the corresponding class. A typical parametric system is the combination of Hidden Markov Model (HMM) and Gaussian Mixture Model (GMM), which assumes Gaussian distribution of each feature in the feature vector.

3.4 Posture and Gesture

Based on our overall system approach we treat the human body posture recognition as a basic classification task. Given a novel binary object mask to be classified, and a database of samples labeled with possible body postures, the previously described shape descriptors are extracted and the image is classified with a chosen classifier. This can be interpreted as a database query: given a query image, extract suitable descriptors and retrieve the best matching human body posture label. Other similar queries are possible, resulting in a number of useful applications, such as skeleton transfer, body posture synthesis and figure correction.

3.5 HMM and GMM

An approach for action recognition by using Hidden Markov Model (HMM) and Gaussian Mixture Modelling (GMM) to model the video and audio streams respectively will be proposed. HMM will be used to merge audio-visual information and to represent the hierarchical structure of the particular violence activity. The visual features are used to characterize the type of shot view. The audio features describe the audio events within a shot (scream, blast, gun shots). The edge, motion (orientation and trajectory) information is then input to a HMM for recognition of the action.

Time series motion data of human’s whole body is used as input. Every category of target action has a corresponding model (action model), and each action model independently calculates the likelihood that the input data belongs to its category. Then the input motion is classified to the most likely action category. The feature extraction (position, direction, movement) focuses attention on the typical motion features of the action, and a model of the features’ behaviour in the form of HMM.

A motion data is interpreted at various levels of abstraction. The HMM expresses what the action is like by symbolic representation of time-series data. In this work, we combine information from features that are based on image differences, audio differences, video differences, and motion differences for feature extraction. Hidden Markov models provide a unifying framework for jointly modeling these features. HMMs are used to build scenes from video which has already been segmented into shots and transitions. States of the HMM consist of the various segments of a video. The HMM contains arcs between states showing the allowable progressions of states. The parameters of the HMM are learned using training data in the form of the frame-to-frame distances for a video labeled with shots, transition types, and motion.

For each type of violent activity (punch and kick), a HMM will be build to characterize the action and interaction processes as observation vectors. A HMM for each action is trained with the corresponding training MPEG video sequences (kick, punch, run, walk, stand, etc.). The expectation maximization (EM) and GMM approaches are then used to classify testing data using the trained models. Once the mean and covariance of the Gaussian model of the training audio data are obtained, the likelihood ratio between the input audio track and the sound classes, is computed to determine which class the associated sound belong to.

4. Experimental Discussion

By using these processes for action detection, the result of successful detected action is depicted in Table 1. As is seen, sitting is characterized as the most significant motion with 80% of success rate. This is due to less motion activities involved.

TABLE 1 Classification of the individual action sequences

Type of Sequence	Total Number	Correctly Classified	% Success
Standing	4	3	75
Sitting	5	4	80
Walking	3	2	67
Punching	5	3	60
Falling	4	3	75
Kicking	6	3	50
Running	2	1	50

These actions were taken from the dataset itself. For example, most of the standing and walking scenes were collected from movie I, Robot. Most of the falling scenes were captured from Rush Hour, sitting and punching from Charlie's Angel, and kicking and running from Matrix. Figure 4 shows some scenes that demonstrate these actions for classification.

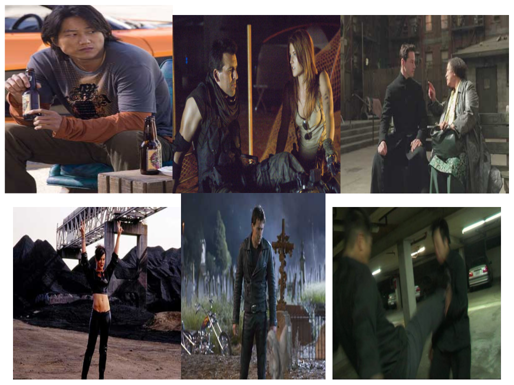


Figure 4.0 Some examples of video clips

5. Conclusion

In this research, we studied on techniques for extracting meaningful features that can be used to extract higher level information from video shots. We will use motion, colour, and edge features, sound and shot activity information to characterize the video data.

The proposed algorithm is implemented in C++ and it works on an Intel Pentium 2.56GHz processor. As described above HMMs are trained from falling, walking, and walking and talking video clips. A total of 64 video clips having 15,823 image frames are used. Some image frames from the video clips are shown in Fig. 4. In all of the clips, only one moving object exists in the scene**.**

In summary, the main contribution of this work is the use of both audio and video tracks to decide an action in video. The audio information is essential to distinguish an action from a person rather than a person simply sitting down or sitting on a floor. To prove the usefulness of the proposed method, the experiments were performed to evaluate the detection performance with several video genres. The experimental results show that the proposed method to detect action scenes gives high detection rate and reasonable processing time. The action detection time was calculated for the case with the multimodal feature set. It takes about 102 seconds to detect action within single video clip with PC (Pentium IV CPU 2.40GHz). The overall process time depends on various factors: CPU clock speed, the type of language used in system implementation, optimization scheme, the complexity and the number of processing steps, etc. Because the proposed action detection is performed with unit of short video clip, the detection time is not affected by the length of entire video.

6. References

1. L. Zelnik-Manor and M. Irani, “Event-based Analysis of Video”. *Proceedings of IEEE Conference Computer Vision and Pattern Recognition*, 2001.
2. C. Stauffer and W.E.L. Grimson, “Learning Patterns of Activities using Real-Time Tracking”. *Journal of IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol (22), no. (8), pp. 747 – 757, 2001.
3. A.A. Efros, A.C. Berg, G. Mori and J. Malik, “Recognizing Action at a Distance”. *Proceedings of International Conference on Computer Vision*, 2003.
4. S. Fischer, R. Lienhart and W. Effelsberg, “Automatic Recognition of Film Genres”, *Proceedings of ACM Multimedia*, pp. 295 – 304, 2003.
5. G. Tzanetakis and P. Cook, “Musical Genre Classification of Audio Signals”, *IEEE Trans. On Speech and Audio Processing*, vol. 10, no. 5, pp. 293 – 302, 2002.
6. C.P., Tan, K.S. Lim, and W.K. Lai. 2008. Multi-Dimensional Features Reduction of Consistency Subset Evaluator on Unsupervised Expectation Maximization Classifier for Imaging Surveillance Application. *International Journal of Image Processing*, vol. 2(1), pp. 18-26.
7. J. P and P.S. Hiremath. 2008. Content Based Image Retrieval using Color Boosted Salient Points and Shape features of an image. 2008. *International Journal of Image Processing*, vol. 2(1), pp. 10-17.

MAHI: Machine And Human Interface

Bhupesh Kumar Singh kumar.bhupesh04@gmail.com
Department of Computer Science & Engineering
Lingaya's University,
Faridabad: 121002, India

G. Sahoo drgsahoo@yahoo.com
Department of Computer Science & Engineering
Birla Institute of Technology, Mesra,
Ranchi: 835 215, India

B.L.Raina rbushan@rediffmail.com
Department of Information & Technology
Lingaya's University,
Faridabad: 121002, India

ABSTRACT

Sketch recognition systems used for the development of many domains are time consuming. This is because it involves intricacies in handling the data with greater care for each domain. Present authors introduced *A new approach to sketch recognition using Heuristic* [1]. It compares very well with the statistical approach used in Bayesian networks[4] In this paper, we have introduced MAHI: *(Machine and Human Interface)* as a sketching language based on geometrical rules associated to the recognition of basic elements in its domain as the better option discussed in MAHII (Machine And Interactive Interface [2]. The recognition engine interprets the sketches based on the information obtained from the description of domains associated with the input from the user.

Keywords: MAHI, MAHII, LADDER, Domain-Description, Geometrical Constraints, Multi-connected lines.

1. INTRODUCTION

Sketch recognition is an important part of sketch-based system and it needs domain information to classify and recognize the shapes. Over the years, sketch recognition systems on pen-based input devices and hand-drawn diagrammatic domains are discuses in details. Stiny and Gips,[5] have given the grammars for shape description language. Jacobi et. al. [6] gave software module that lacked improvement to code the domain-dependent recognition system. Bimber et. al [7] introduced BNF grammar with the restriction to only shape information and provided no other helpful information. Mahoney [8], Caetano et. al. [9] use languages (including Fuzzy related grammar) were again subject to some limitation.

Recently T.Hammond & R.Davis [10][11][12][14][16] have discussed LADDER as a first sketching language. The author's claim that the language was designed and developed successfully while describe its domain classes. Unfortunately the application of language is subject to a few limitations of describing shapes such as a fixed graphical grammar, shapes that have a lot of regularity and not too much of the details. The language can describe domains for a few curves and there is difficulty in specifying curves control points.

To reduce the limitations of language a new approach to sketch recognition has been proposed in [1].Recently we have introduced its interface MAHII in [2]. In this paper, we describe a language that provides the description of basic shapes of sketch in terms of its geometry, defining the domains that implicitly inherit the geometrical based shapes. Since the dependence of domain sketch recognition system is the main disadvantage for all the systems, therefore to provide a natural and more attractive interface [2], MAHI places minimum constraints on the users in view of the system being independent of the domain.

MAHI is the language describing the shapes drawn, displayed, and edited subject to the domain. It has been successfully tested to multi-domain recognition system along with a code generator that parses MAHII interface based on domain description and generates the codes pertaining to MATLAB to give complete recognition to the system.

MAHI identifies a few basic shapes through which recognition engine can identify its corresponding domains and then it connects the particular domain to itself through an interfacing. This way, the system provides the developer to define the domains independently which describe what the domains shapes look like, and how they should be edited and displayed after they are recognized. The recognition engine is followed by heuristic engine that finally recognizes the input sketch completely.

We discuss components of MAHI (language), contents of the sketch grammar in MAHI, Testing, and System- Implementation alongwith conclusion in the subsequent sections.

2. MAHI LANGUAGE

MAHI domain descriptions include defined shapes and other information related to the recognition process, such as stroke order and stroke gradient. It contains the defined geometrical shapes with minimum constraints and takes into account its editing behavior, display methods and syntax for specifying a domain description. We create a domain description reusing defined shapes to build the desired shape hierarchically.

2.1 ELEMENTS OF MAHI

In MAHI (figure 1) we define the geometric elements in terms of the entities given: The basic core entity (BCE) unit defines a *point,* a collection of BCE forms a core entity (CE) defined by a line or an arc and the collection of BCE and CE forms a derived entities (DE) as open or closed shaped.

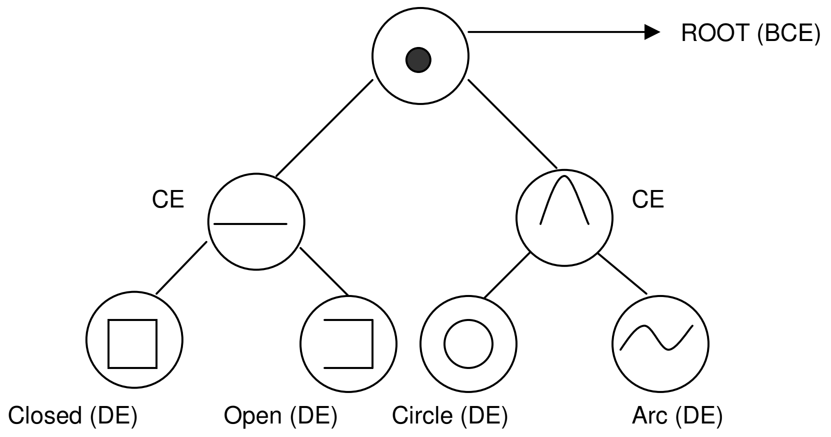


FIGURE 1: Basic Elements

2.2 GEOMETRICAL ANALYSIS

In MAHI, we are able to draw any of the shapes based on entity BCE, CE, and DE as defined by *point, arc, line, rectangle, square, circle, ellipse, polygon, curves, surfaces, volume etc*. Here a shape is a geometrical structure with basic input as geometrical rules associated to recognition process such as the stroke. Using the shape properties, an arbitrary shape is extended or reduced to the desired shape. An application designer can redefine the properties as and when he needs. The language has proven to be very powerful and highly interactive for multi domains. The language enables more accurate and fast sketch recognition by using bottom-up as well as top-down recognition in view of mathematical definition of line being defined as locus of point and point being defined as limiting case of a line. Thus, a point is recognized and is used to identify and be identified as a line or arc since the line or arc is extended to form any other extended shape(s) or extended shape(s) merged to a line or curve.

2.3 SHAPE-GEOMETRY

Generally, we define a point as a precise location or place on a plane, usually represented by a dot. We remark here that geometrically it is appropriate to define a point in terms of line segment as such; we first define a line segment as given by the following definition:
We define a line segment as the shortest distance between any two given points, whereas a point is a line segment of join of two points whose distance in one dimensions or gradient in two dimensions tend to zero. Further, if the distance between two consecutive points is not minimum then it will define an arc and consequently piece-wise continuous arcs will give rise to a curve.

- Shapes in 2 & 3 Dimensions

  Shape of point in 2 Dimensions: Given two points $P(x_1,y_1)$ & $Q(x_2,y_2)$ we associate metric d joining P & Q such that $d(\overrightarrow{PQ})\to 0$, giving $x_1\to x_2, y_1\to y_2$. i.e. $P(x_1,y_1)\to Q(x_2,y_2)$ implies that the limiting case of a line defines a point. Similarly $P(x_1,y_1,z_1,)\to Q(x_2,y_2,z_2)$ gives a point in three dimensions.

- Arc in 2 & 3 Dimensions

  If A & B are any two points in a plane with regard to point O as the origin, then as per mathematical definition we define length of Arc AB given by: $\widehat{AB} = |OA| \quad \angle\theta$ corresponding to the equation of the curve r=f(θ) where r= $|OA|$ & $\angle\theta = \angle AOB$ .Same results hold true in 3-dimensional space.

Using entities defined by BCE, CE or DE given by figure 1 we obtain any shape (linear or curve linear). Further, these entities can give rise to any of the shapes in different domains such as states in finite state machine, formation of pulley in mechanical engineering, circuit design in electronic engineering, etc.

2.4 SHAPES WITH CONSTRAINTS

A number of geometrical shapes can be defined by the algebraic equations subject to the given constraints. For example ax+by+c=0 subject to the condition $d \le x \le e$ represents a number of lines including family of parallel, perpendicular lines etc. in a plane of two dimensions for different values of a, b, c, d and e. In other words if the sketch grammar consisting of algebraic equation subject to the constraints as given above is represented geometrically, then we find that it is far superior to express constraints in terms of mathematical language than otherwise given by traditional constraints such as rotatable, angle, horizontal, vertical, etc. as defined in [9][15] .

The authors [10][11][15] presented a language containing a library of pre-defined shapes with pre-defined constraints such as *horizontal, posSlope, above left* etc including *IsRotable, angle L, vertical constraints* to define the orientation subject to the relative co-ordinate system, which seems cumbersome to the user. However, MAHI ignores all these complicated expression and having inbuilt mathematical system, the shape takes the form with ease than in Ladder. In MAHI gradient for any line in two dimensions is given by m_i= -(coefficient(x))/(Coefficient(y)), i=1,2 as such a user can draw any line following the language which naturally is one of these: Perpendicular, parallel, collinear, coincident, same side, opposite side meet, intersect, tangent, centre below, centre above, positive slope, negative slope, etc. The components and properties of these lines can be used hierarchically bottom-up or top-down in the shape descriptions. The other related new shapes are drawn by extending the properties of a line, rectangle, parallelogram, circle, to any of the shapes. Thus, in general, a new shape defined by the components, geometrical constraints, alasis, etc. as given in figure 2:

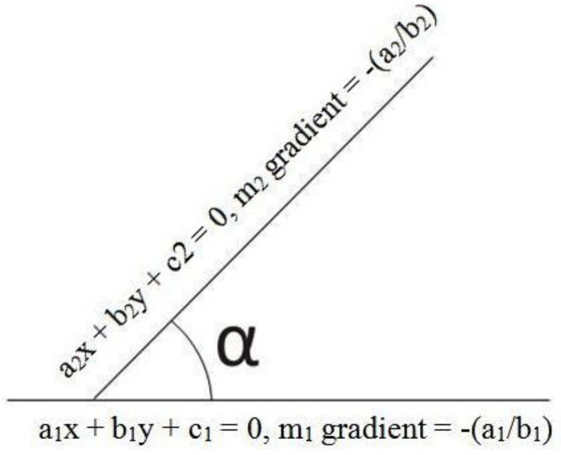

$a_2x + b_2y + c2 = 0$, m_2 gradient = $-(a_2/b_2)$
α
$a_1x + b_1y + c_1 = 0$, m_1 gradient = $-(a_1/b_1)$

FIGURE 2: Geometrical Constraints

2.5 DEFINED-EDITING BEHAVIOR

MAHI knows when and how to recognize, edit and display the shapes. Editing and displaying are the important components of sketch interface that vary as per different domains. The display of the object helps the sketcher to make the recognition not only possible-shape but makes it beautiful by removing the clutters followed by editing gestures consisting of event for each shape. The advantage in MAHI is that a user is encouraged to standardize different domains by including some defined editing behavior. In fact, one can define one's own editing behaviors for each domain. *For example: Using the same gesture such as drawing a point inside a circle may be intended as a check in one domain(check box) and a full stop in the other domain (text box) or as an editing command .* We alternatively define the above definition in section I

```
Edit ( <number of edges or segments 'n'> and <information about each segment>)

Loop ( for i = 1 to n )
(
        trigger DoubleClickHoldDrag side 'i'
        action
        translate this
        setCursor DRAG
        showHandle MOVE side 'i'
        // where 'i'  is the side number

        trigger holdDrag side 'i'
        action
        translate this
        setCursor DRAG
        showHandle MOVE side'i' … side'n'
)
```

FIGURE 3: Editing Behavior

2.6 CURSER BEHAVIOR

Use of Editing (curser) in MAHI is dependent on its mode that, one can use pen based editing that is when the curser is in pen mode (sketching) in which case the user can draw sketches and when it is in curser mode it means the editing mode. Thus, sketching and editing use distinct pen motions. One of the editing behaviors in MAHI is if one clicks and holds the pen on the surface of the rectangle and drags the pen, the entire rectangle will translate along with the movement of the rectangle. In this way the rectangle along with the vector is translated, scaled, and rotated as one whole shape. All of editing behaviors also change the pen cursor as display to the sketcher to know of his performing an editing command. MAHI editing behavior of triggers include *click, double click, hold, hold drag, encircle, etc*. along with their subsequent application.

2.7 DISPLAY METHODS

Controlling is a vital part of sketching interface when the user recognizes the shape after he has drawn that sketch. MAHI defines recognition layer that has the power of the recognition of the sketch. It gets the input from editing layer as defined above and uses heuristic engine to perform semantic checking for the relevant details from the system database. It also includes the concept of domain independence with reference to multi-domain. It recognizes the output and displays the results to the input/output layer of MHAII interface [2].

2.8 VECTOR

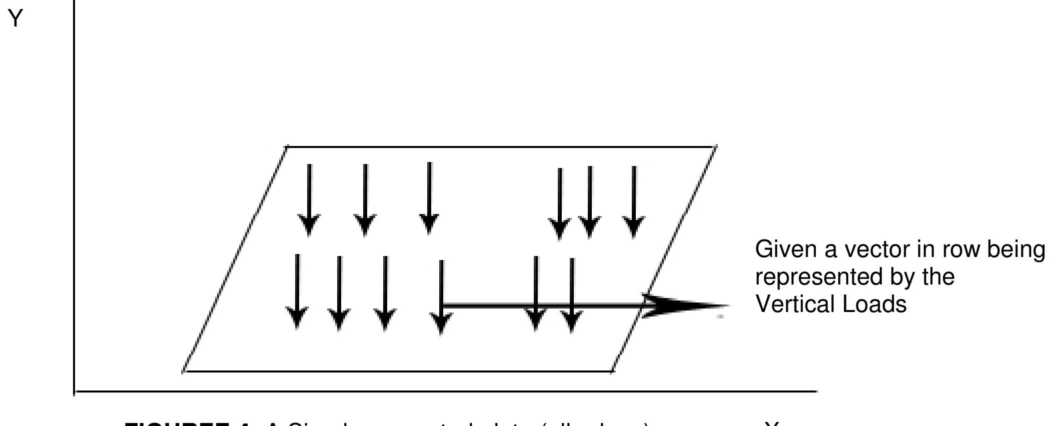
It is defined as a quantity having magnitude and orientation. The magnitude of vector is denoted by $|AB|$ where AB is the length of line segment joining points A&B and orientation is given by the arrowhead $\rightarrow$. Thus, a vector is defined by $\overrightarrow{AB}$. Our discussion with regard to the rectangle given by ABCD is such that $\overrightarrow{AB} = \overrightarrow{CD}$, $\overrightarrow{AD} \perp \overrightarrow{AB}$ as shown in figure 5.

2.9 SCIENTIFIC & ENGINEERING APPLICATION OF SHAPE

We use multi- domain shapes to provide context so that we may more effectively recognize shapes in various domains. A user may specify that when we move a rectangular plate along with the arrow of heads acting on this plate, the arrow should move with the rectangle as shown by the figure 4. For example, in Engineering, we are confronted with the problem of simply supported rectangular plate subject to uniform load governed by the differential equation given by:

$$\frac{Eh^3}{12(1-v^2)}\left(\frac{\partial^4}{\partial x^4}+2\frac{\partial^4}{\partial x^2 \partial y^2}+\frac{\partial^4}{\partial y^4}\right)w - q_o = 0 \text{ ----------------------------------(1)}$$

Where E, h, v, q_o and w have the usual meanings.


Y
Given a vector in row being represented by the Vertical Loads
X

FIGUREE 4: A Simply supported plate (all edges)

We further remark here that in reference to editing behavior as defined above, the display of vertical arrow as given in figure 4 and 5 have different meanings such as in figure 4 it represents the load acting on the plate in one domain (Three dimensions) and vertical force in the other domain(Two dimensions).

3. DOMAIN DESCRIPTION OF MAHI

We define domain description as an aggregate of elements. Here elements are referred as the list of domain shapes and group shapes.

3.1 MAHI SHAPE-DEFINITION

A shape is defined if it is associated to a domain belonging to domain description. In particular, geometrical shapes are defined as the blocks belonging to the domain shapes of multi-domain.
Now using the above definition alongwith the basic entity as defined by BCE, CE, DE in MAHI given in figure1, we can form new desired shape as:

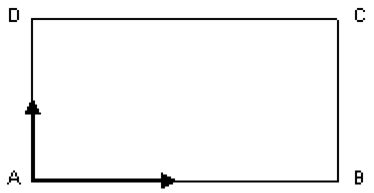


FIGURE 5: Rectangle

Formation of figures 5 is obtained by any of the following two rules:

Rule 1: Description of rectangle ABCD as locus of a point.

The locus of a point A to B horizontally & B to C vertically & C to D horizontally & D to A is vertically is a rectangle ABCD.

Rule 2: Description as inputs of rectangle based on line segment, geometrical constraints, aliases etc.

- In this case, we list the ***components such as AB,AD,DC&CB*** as the side of a rectangle ABCD.
  Equations: of two parallel lines AB and DC are given by
  $a_ix+b_iy+c_i^k=0;i=1,k=1,2$ ....(2) (by section 2.4)
  Equations of other two parallel lines AD and BC are given by
  $b_ix-a_iy+c_i^{k'}=0;i=2, k'=1,2$......................................(3), $k,k' \in R$, since $\overrightarrow{AD} \perp \overrightarrow{AB}$
- The components of the above rectangle are drawn subject to ***geometrical constraints:*** given, as gradient of AB is equal to negative reciprocal of the gradient of line AD. i.e. (gradient of AB)*(gradient of line AD)= -1, here gradient of a line (2 or 3) = -(coefficient(x)) /(coefficient(y))

- Alternative to the above expression of perpendicularity of the lines are also be defined as a set of ***aliases (orientation)*** *for* drawing of a rectangle. In this case, the head and tail have been added as aliases in the arrow definition given by $\overrightarrow{AB} \perp \overrightarrow{AD}$ as shown in figure 5 using specifying editing behaviors.
- Hierarchical shape definition: Four lines with two each of the vectors with a pair of parallel lines for rectangle ABCD of Figure 5. as given by $\overrightarrow{AB} = \overrightarrow{DC}$ and $\overrightarrow{AD} \perp \overrightarrow{AB}$

- MAHI defines the editing layer, which edits the sketch input. This layer includes {Pre-processing of sketch [3] and provides the input to the recognition layer. the functioning of image filtering (Gaussian FIR filter [3]), image segmentation (Canny Method [13] [15]) and image editing [3]. It provides in input to the recognition layer.

- Editing gestures permit us to recognize the system to differentiate between sketching and editing. MAHII language includes a number of predefined editing behaviors using the algebraic constraints associated to the geometry of the figures such as a point, line, arc, etc. The possible editing action include *wait, select, deselect, delete, translate, scale, IsRotate* etc.
- Editing behaviors specifies the gesture of dragging the component in any direction. In this case, the actions of these editing commands specify that the object should follow the pen at the description of the user including translating and rotating. Any arbitrary quadrilateral can be changed to a rectangle as per the action of the user, resulting in the change of coefficient of x and coefficient of y in the domain.
- Display methods indicate what to display when the object is recognized. The sketching shape follows the original component followed by the constraints associated to the other related component to give rise to the extended shape such as line, circle, rectangle, etc. to give the final shape.

The MAHI (Sketching Language) based on the domain description is translated into shape recognizer such as (geometrical) components & constraints. MAHI has the exhibitors and editors representing the display and edit sections respectively to be used in conjunction with the recognition systems to obtain a drawn sketch.

```
define shape Rectangle
description "A quadrilateral with four sides and all angles 90 degree”

Components
Line side1
Line side2
Line side3
Line side4

Clauses          // Constraints
coincident side1.p1 side2.p1
coincident side2.p2 side3.p2
coincident side3.p3 side4.p3
coincident side4.p4 side1.p4

parallel side1 side3
parallel side2 side4
perpendicular side1 side4
perpendicular side1 side2
perpendicular side2 side3
perpendicular side3 side4

equalLength side1 side3
equalLength side2 side4

Aliases

Point side1 side2.p1
Point side2 side3.p2
Point side3 side4.p3
Point side4 side1.p4

Editing

Edit ( side1 side2 side3 side4)
display original-strokes
```

FIGURE 6: DOMAIN-DESCRIPTION OF RECTANGLE (MAHI)

4. TESTING

In domain description, we have defined MAHI as the language being described by a number of symbols belonging to the various domains. MAHI has been successfully applied to around 50 shapes from flow chart and several other domains of engineering or non-engineering.

4.1 MULTI-CONNECTED LINES

Polyhedron is defined in three or higher dimensions, and polygon is defined as a plane shape in 2-dimensions. A polygon is defined as the connected number of line segments and these are formed as the extension of two lines meeting at a point being defined as co-initial bi-lines. Bi-lines form the basis for the extension of polyLines. However, we can draw piece-wise poly-linear or arcular segments as shown in figure 7.

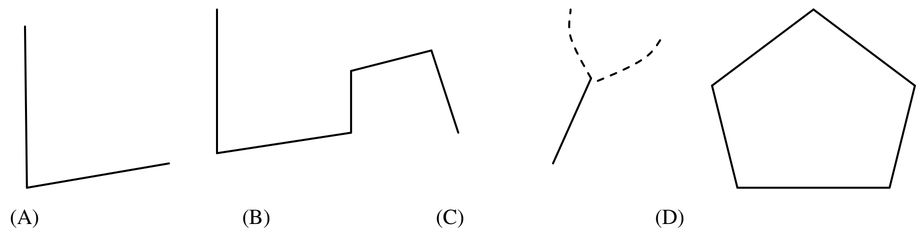


FIGURE 7 : (A) Bi-lines (B) Multi Open Lines (C) Union of lines Segment and Piece – wise arcs, (D) Pentagon

```
define shape MultiLine
description "A set of intersecting lines"

Components
Line side1
Line side2
Line side3
…
Line siden

Clauses         // Constraints
coincident side1.p1 side2.p1
coincident side2.p2 side3.p2
coincident side3.p3 side4.p3
…
coincident side(n-1).p4 siden.p4

Aliases

Point side1 side2.p1
Point side2 side3.p2
Point side3 side4.p3
…
Point side(n-1) side(n).p(n-1)

Editing

Edit (side1 side2 side3 … side (n))

display original-strokes
```

FIGURE 8: Description of MultiLine

```
define shape Bi-Line
description "A pair of intersecting lines"

Components
Line side1
Line side2

Clauses          // Constraints
coincident side1.p1 side2.p1
coincident side2.p2 side3.p2

Aliases

Point side1 side2.p1
Point side2 side3.p2

Editing

Edit ( side1 side2)

display original-strokes
```

FIGURE 9: Description of Bi-Lines

```
define shape Pentagon
description "A plane figure with five straight sides and five angles"

Components
Line side1
Line side2
Line side3
Line side4
Line side5

Clauses          // Constraints
coincident side1.p1 side2.p1
coincident side2.p2 side3.p2
coincident side3.p3 side4.p3
coincident side4.p4 side5.p4
coincident side5.p5 side1.p5

Aliases
Point side1 side2.p1
Point side2 side3.p2
Point side3 side4.p3
Point side4 side5.p4
Point side5 side1.p5

Editing

Edit (side1 side2 side3 side4 side5)

display original-strokes
```

FIGURE 10: Description of Pentagon

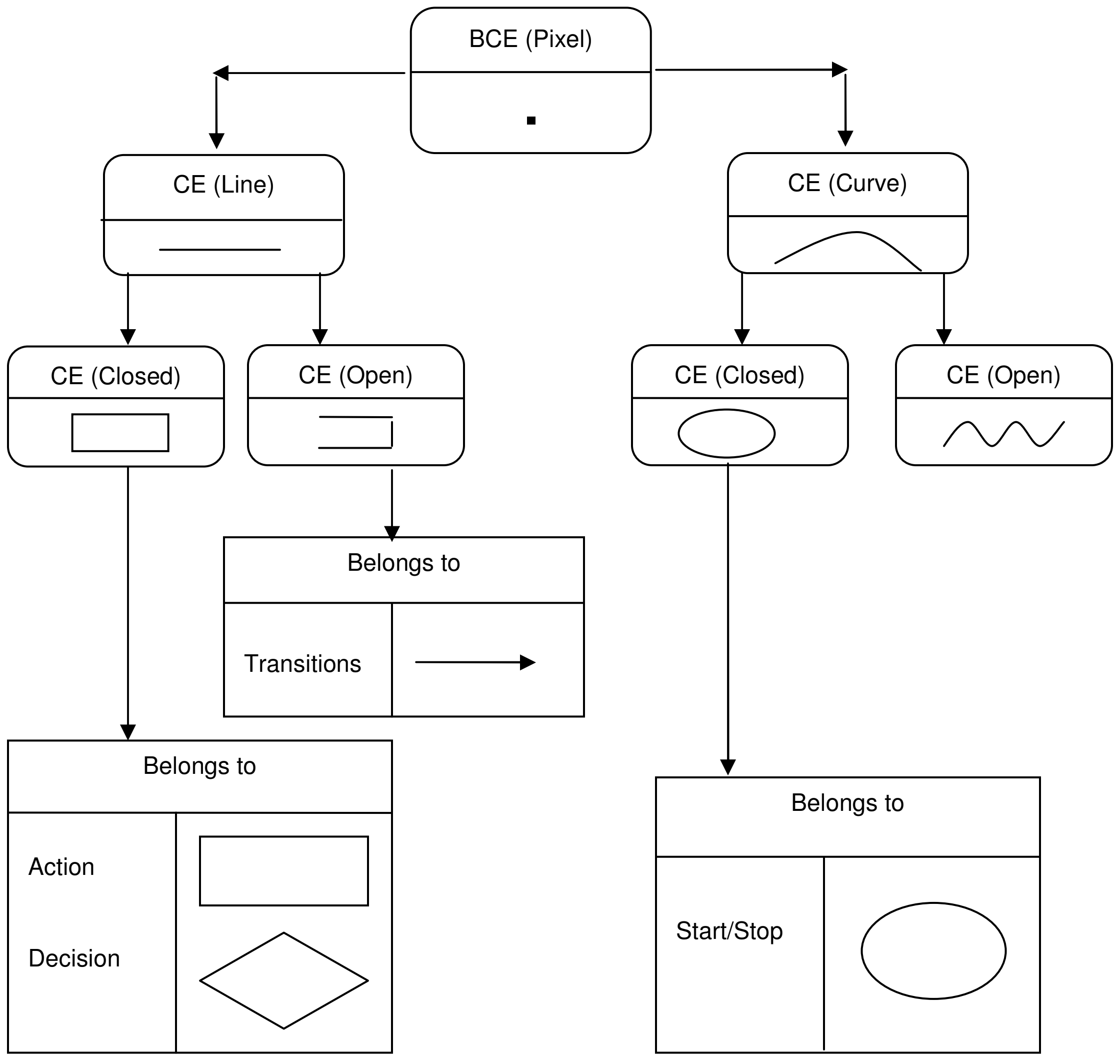


FIGURE 11: Inheritance diagram of Flow Chart diagram

4.2 SYSTEM IMPLEMENTATION

We have drawn the sketches and tested the same to confirm whether these were recognized by our domain-independent recognition.

We built a simple domain-independent recognition system to test whether sketches is recognized from our domain descriptions. The system parses a domain description into MATLAB code and Jess (a rule-based system that interface Java) [Friedman-Hill, 1995] rules, and uses them to recognize sketches. Using the domain description of MAHI, the system successfully recognized hand-drawn sketches as rectangle for any domains (Engineering & Non-Engineering).

Thus any hand drawn sketch can be recognized by parsing to various domains of MAHI.

4.3 PARSING

The domain description MAHI is parsed to produce recognition code, generating one Jess rule (recognition information), and one MATLAB file (shape description), for each shape description. The system uses the Jess rules to recognize sketches. For example figure 5 is shown in figure 12.

```
(defrule RectangleCheck
;;Get the parts of the rectangle
?f0 <- (Subshapes Sides ?s $?s_list)
(Coincident ?s ?s_side1 ?s_side2 ?s_side3 ?s_side4)
(Line ?s_side1 ?side1_p1 ?side1_p2)
(Line ?s_side2 ?side2_p1 ?side2_p2)
(Line ?s_side3 ?side3_p1 ?side3_p2)
(Line ?s_side4 ?side4_p1 ?side4_p2)
;; test for that the coincident edges
(test (coincident ?s_side1_p1 ?s_side2_p1))
(test (coincident ?s_side1_p1 ?s_side2_p2))
(test (coincident ?s_side2_p1 ?s_side3_p1))
(test (coincident ?s_side2_p1 ?s_side3_p2))
(test (coincident ?s_side3_p1 ?s_side4_p1))
(test (coincident ?s_side3_p1 ?s_side4_p2))
(test (coincident ?s_side4_p1 ?s_side1_p1))
(test (coincident ?s_side4_p1 ?s_side1_p2))
;; test for the parallel edges
(test (parallel ?s_side1 ?s_side3))
(test (parallel ?s_side2 ?s_side4))
;; test for the perpendicular edges
(test (perpendicular ?s_side1 ?s_side2))
(test (perpendicular ?s_side2 ?s_side3))
(test (perpendicular ?s_side3 ?s_side4))
(test (perpendicular ?s_side4 ?s_side2))
;; test for the equal edges
(test (equal ?s_side1 ?s_side3))
(test (equal ?s_side2 ?s_side4))

=>
;; Rectangle found successfully
;; Set the aliases
(bind ?side1 ?oa_side1) (bind ?side2 ?oa_side2)
(bind ?side3 ?oa_side3) (bind ?side4 ?oa_side4)

;; Notify recognition system that a Rectangle is recognized
(bind ?nextnum (addshape Rectangle ?s $?s_list ?side1? side2? side3? side4?))
;; Tell the Jess system that a Rectangle is recognized
(assert (Rectangle ?nextnum ?s ?side1 ?side2 ?side3 ?side4 ?))
;; Rectangle is a domain shape. Assert it.
;; Conflicts will be resolved elsewhere.
(assert (DomainShape Rectangle ?nextnum (time)))
```

FIGURE 12: Automatically generated Jess Rule for the Rectangle.

5. CONCLUSION

MAHI language is primarily based on shape and the domain description and can include any type of information that could be helpful to the recognition process. It consist of pre-defined basic shapes alongwith the constraints & editing behaviors for specifying the domain description. Interestingly its mathematical description, apart from describing how sketch diagrams in a domain are drawn, displayed and edited allows the shapes to obtain new shapes hierarchically. Its shape group describes how domain shapes interact and provide information in top-down as well as bottom–up recognition. We are also able to build a simple domain-independent mathematical-rule based sketch recognition system, that tests if the recognition is viable.

ACKNOWLEDGEMENT :

We owe our sincere thanks to Prof. G.V.K. Sinha (Gadde), Chancellor and Dr. Y.S. Goel Director/Principal, Lingaya's University , Faridabad for their keen interest in our work.

6. REFERENCES

[1] G.Sahoo & Bhupesh Kumar Singh, "*A New Approach to Sketch Recognition using Heuristic"*, International Journal of Computer Science and Network Security, Vol. 8 No. 2, 2008., PP. 102-108.,

[2] G.Sahoo & Bhupesh Kumar Singh, *"MAHII :Machine And Human Interactive Interface"* International Journal of Image Processing, Volume (2), Issue (3), 2008 PP. 1-10.,

[3] G.Sahoo & Bhupesh Kumar Singh *"A Human Detector and Identification System"* International Journal of Computer Science, Systems Engineering and Information Technology volume Vol.1(1), June 2008. PP. 39-44

[4] Chien C.F., *"Modifying the inconsistency of Bayesian networks and a comparison study for fault location on electricity distribution feeders"* , International Journal Operational Research, Vol. 1, Nos. ½, pp. 188-202, 2005.

[5] Stiny & Gips, *".Shape grammars and the generative specification of painting and sculpture"* Information Processing, pages 1460–1465, 1972.

[6] Jacob *et al.*, & S Morrison, *"A software model and specification language for non-WIMP user interfaces"* ACM Transactions on Computer-Human Interaction, 6(1):1–46, 1999.

[7] Bimber *et al.*, LM Encarnao, & A Stork, *"A multi-layered architecture for sketch-based interaction within virtual Environments"*. Computer and Graphics, 2000

[8] J.V. Mahoney & M.P.J. Fromherz, *"Three main concerns in sketch recognition and an approach to addressing Them"* AAAI Spring Symposium on Sketch Understanding, PP. 105–112, March 25-27 2002.

[9] A Caetano, N Goulart, M Fonseca, & J Jorge. *"Javasketchit: Issues in sketching the look of user interfaces"* AAAI Spring Symposium on Sketch Understanding, 2002.

[10] Tracy Hammond & Randall Davis. *"LADDER: A language to describe drawing, display, and editing in sketch recognition"* In Proceedings of the 2003 International Joint Conference on Artificial Intelligence (IJCAI-03), Acapulco, Mexico, 2003.

[11] Tracy Hammond and Randall Davis. LADDER, a sketching language for user interface developers. Elsevier, Computers and Graphics, 28:518–532, 2005.

[12] Tracy Hammond & Randall Davis, "LADDER: A Perceptually-based Language to Simplify Sketch Recognition User Interface Development" MASSACHUSETTS INSTITUTE OF TECHNOLOGY January 2007

[13] Jain, A.K., "*Fundamental of Digital Image Processing*", Prentice Hall. *University of California-Davis.,* 1989

[14] S.J.B. Shum, A. MacLean, V.M.E. Bellotti, and N.V. Hammond. Graphical argumentation and design cognition. *Human-Computer Interaction,* 12(3):267-300, 1996.

[15] Michael Shilman, Hanna Pasula, Stuart Russell, and Richard Newton. Statistical visual language models for ink parsing. In *Sketch Understanding, Papers from the 2002 AAAI Spring Symposium,* pages 126-132.,

[16] Tevfik Metin Sezgin, Thomas Stahovich, and Randall Davis. Sketch based interfaces: Early processing for sketch understanding. In *The Proceedings of 2001 Perceptive User Interfaces Workshop (PUI'01),* Orlando, FL, November 2001.