

Head Tracking System for Games and Simulations

Orlando HERNANDEZ and Daniel MARZULLO

Department Electrical and Computer Engineering, The College of New Jersey
Ewing, New Jersey 08628-0718, USA

ABSTRACT

A system is presented which provides head tracking services that can be implemented into 3D games and simulations. This creates an enhanced illusion of depth based on the real world position of the user and gives the ability to look around the virtual world by physically turning your head. A webcam is used to capture images of a layout of three red dots attached to a hat worn by the user. These images are then analyzed by software using blob detection to locate the position of the three red dots. These 2D positions are then used to calculate the 3D position and orientation of the user. This data is then finally communicated to an external game or simulation for viewpoint adjustment.

Keywords: Head Tracking, Virtual Reality, Computer Vision

1. INTRODUCTION

With modern day computer and graphics technology advancing at such an incredible pace [1 - 4], modern video games are becoming increasingly realistic. Despite the realistic visuals however, recent games have not been able to significantly increase the level of immersion that the player experiences. Since the creation of video games the only existing methods of user input has been a keyboard and mouse or a handheld controller, neither providing much immersion. Although there have been a few attempts at a human-motion controller, the implementations have not met their potential and virtual reality products in general have not yet met consumer level accessibility or pricing.

A practical addition to the traditional controller that will leave the hand-held controller or keyboard and mouse schemes unaffected is a head tracking system. In 3D games, the view that the player has into the game world is simulated through a virtual camera. By tracking the head of the player, actual position in the real world can be translated into the 3D game world through adjustment of the viewpoint of this virtual camera. This provides an enhanced illusion of depth beyond what can be done with 3D to 2D projections. Additionally, by tracking orientation, the player can look around the virtual game world by physically turning his/her head. Because head movement is independent from hand movement and is also unhindered when sitting down, this additional control method can be applied to nearly all 3D games and simulations and add a convincing virtual reality experience

Specifications

During the design of the project, there were specifications and design goals that needed to be met. They are as follows.

1. Adaptive image analysis: Various lighting conditions, people with various heights, and cameras with various image output formats will result in a different environment for analysis.
2. Maintain a smooth frame rate for use with real-time games and simulations.
3. Minimize cost.

2. BACKGROUND

Virtual Reality can be achieved in a variety of ways, each providing various levels of user interaction and immersion. Depictions of virtual reality in science fiction, such as in the movie *The Matrix*, where human brains are connected directly to a computer and completely immersed in a computer generated world, showcase the ultimate goal of research in virtual reality. However, because the technologies for systems like this do not yet exist, virtual reality today must be approximated using mechanical and electronic systems.

In many cases, this may consist of a wearable device that measures body movement. These measurements are then sent to a computer to be used in an interactive simulation displayed for the user. Another popular approach is to use computer vision to provide the user interface to the simulation. This allows for a much more versatile system. Because no proprietary hardware is required, changes to the user interface can be made completely in software and body movement is not limited to hardware limitations. However, there are also drawbacks to the entirely computer vision dependent system. For instance, lighting conditions, limitations in camera speed and resolution, and computer processing performance can all affect the accuracy of the system and limit what kind of experience it can provide. Also, because image data is acquired through a camera, objects that are blocked from view cannot be processed. Computer vision based motion controllers have also recently gained major public attention because of the development of such systems for the 3 major video game consoles. Although these systems are still in the development phase, they promise to introduce virtual reality into millions of homes.

3. PROPOSED ARCHITECTURE

The personal computer has always been associated with gaming, simulations, and 3D in general. Flight simulators and racing simulators, two of the most popular PC applications, even have an extensive choice of joystick and steering wheel peripherals to enhance the realism of the game. However, as most 3D games are viewed from the first person perspective, it would be realistic to be able to use head movements to control the viewpoint in the 3D world.

By using a computer vision system, the head can be tracked and used for interacting with a game. Because of the available PC expansion ports, such as USB, a camera can be easily attached and used for this purpose. Also, unlike video game consoles, the PC platform has a large community of people who modify existing games, sometimes creating whole new ones. This "mod" community is even supported by the game developers, who supply tools and code to help with this process. Because of this, even existing games can be modified to accept the new control mechanism. The block diagram for this system is shown in Figure 1.

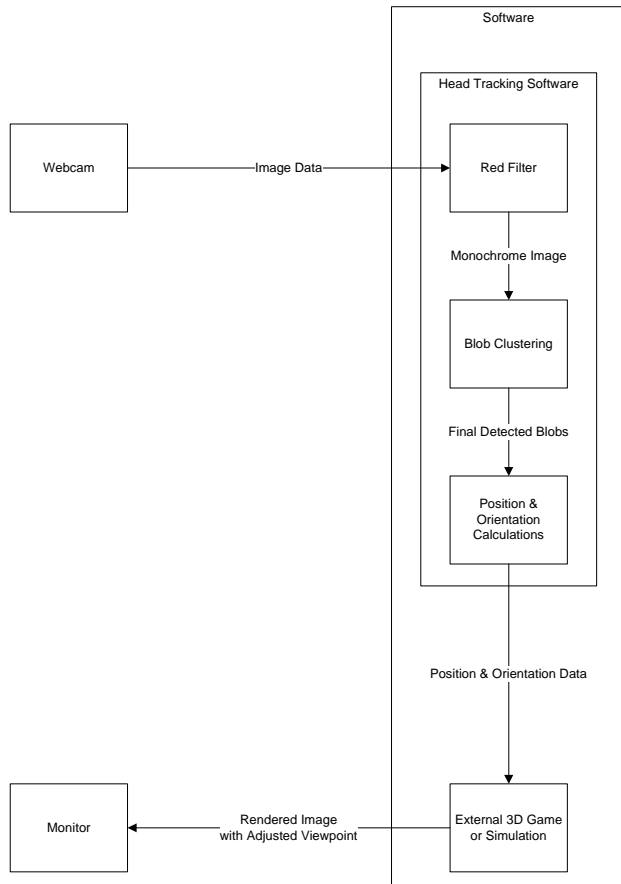


Figure 1: System Block Diagram

4. IMAGE ACQUISITION

Hardware

The design for the head tracking system began with choosing a method to determine the position and orientation of the head based on a captured 2D image. Initial considerations were to have the user wear a dot layout on a hat that could be analyzed to reconstruct a 3D position and orientation. This layout consists of three dots organized into an isosceles triangle pointing upwards. Three dots are used because they allow for all 6 degrees of movement of the head to be determined. Originally all the dots were placed on the same plane in space. However, due to the ambiguity this caused when determining if the head was turned to the right or to the left, and also when determining if the head was turned up or down, the top dot was recessed behind the lower two dots. The parallax effect this new layout provides when turning the head allows the orientation to be uniquely determined. The prototype hat with the dot layout is shown in Figure 2. An updated hat is shown in Figure 3. Any

brimmed baseball hat can be used as a base for the dot layout. The dot layout can be made out of matte household materials. A foam clown nose is ideal for the top dot as it is not reflective and, because it is spherical, will not cause a distorted Z position when the user is looking to the side.



Figure 2: Prototype Hat with Red Dot Layout



Figure 3: Updated Hat with Red Dot Layout

In order to capture images of this dot layout a camera is placed on top of the computer monitor facing the user. A webcam is a convenient solution for the system for many reasons. Most webcams are connected through USB, which enables them to directly communicate with software. They are inexpensive which minimizes cost. Most importantly, many PC owners already own a webcam and have it mounted to the top of their monitors. Many laptops even come with them preinstalled. This drastically increases the base of users with compatible hardware.

Software

One of the goals for the project is for the head tracking system to be compatible with a variety of webcams. In order to achieve this, the head tracking software was based around the Open CV platform. Open CV, short for Open Computer Vision, is a collection of C++ libraries developed by Intel. These libraries include a variety of useful functions that span a wide selection of standard computer vision algorithms. However they also

include supporting functions that help interface with external components such as setting up windows for image viewing and reading mouse input. However the main reason Open CV is used for this project is that it provides functionality to interface with capture devices, including webcams, and access the image data in RGB format. Additionally, the window creation functions are used to view the capture images, the filtered images, and to debug the software. The choice to use Open CV, however, limits the software to be compatible with only the Windows or Linux operating systems. This is not a major concern though as most games and simulations have been designed solely for Windows.

5. IMAGE ANALYSIS

With access to the image data, the position of the dots can now be found. To analyze the image for these dots, a logical image pipeline was designed. This consists of a filtering stage to condition the image for blob detection and a blob clustering stage to group pixels of interest to form the blobs used to determine the location of the dots. Finally, with the three locations of the dots, the position and orientation components are calculated. A figure of this image pipeline is shown in Figure 4.

Initial considerations were to make the dots reflect infrared light, as this would provide a high signal-to-noise ratio that would be easy to filter from the image. However, this would not only require an infrared light source mounted on the monitor, but also require the purchase of an infrared camera. In order to meet the design goal of working with a variety of webcams and to reduce the need to purchase hardware, a new method had to be devised. Instead of infrared reflective dots, red dots were used. By using a color in the visible spectrum, a typical webcam can be used. Additionally, any user can make a red dot layout on a baseball hat using household items.

Filtering

In order to extract the dots from the image, a blob detection algorithm was implemented. First, the captured image, stored in RGB format with three color channels, is converted to a monochrome image with one channel. This conversion process is accomplished by determining if each pixel of the source image is red, and if so, storing the corresponding pixel in the monochrome image as its maximum value. A pixel is determined to be red if the value of its red channel component is higher than the values of both the green and blue channel components by specified thresholds. These thresholds can be individually modified through graphical slider bars. These values can be fine tuned until a strong image of the red dots is produced with little surrounding noise. An example of the output produced by this filtering is shown in Figure 5.

Blob Clustering

Now that there is a binary image formed from the filter stage, the pixels can be clustered to form blobs. This process goes through the image data row by row and finds where a contiguous row of pixels begins and ends. As these rows are found, a new blob may be stored. When a row of pixels is found, it is compared to the rows of pixels found on the previous line. If any of these rows overlap, the new row is added on to the blob that the previous line row of pixels belongs to. The blobs are stored separately from the rows of pixels found on the lines so that the blobs can be easily accessed and organized.

Head Tracking System Image Pipeline

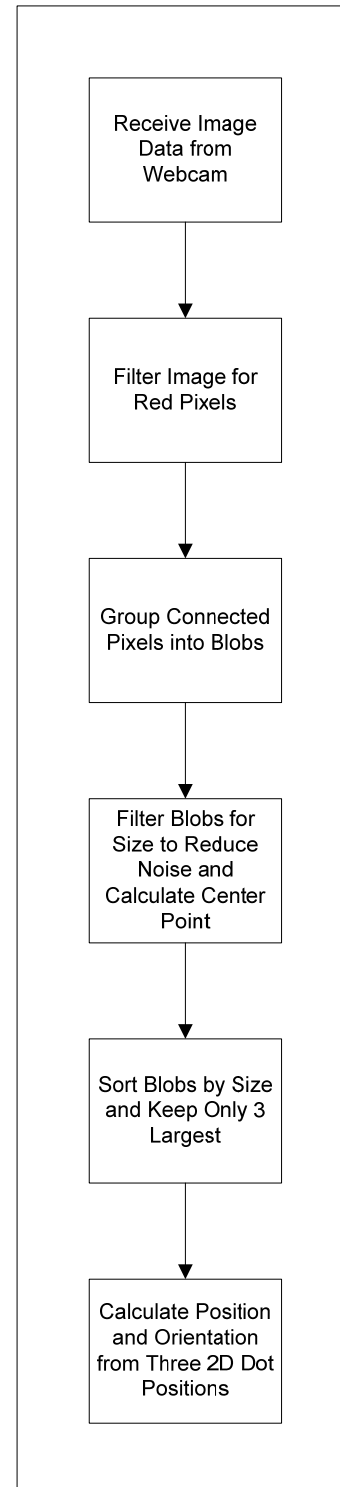


Figure 4: Image Pipeline

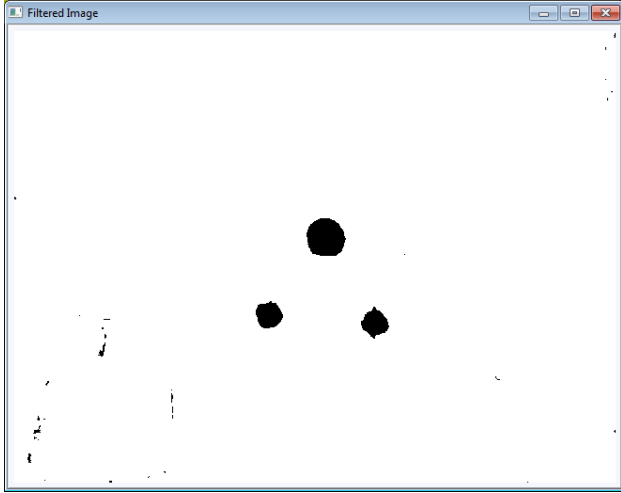


Figure 5: Typical Filtered Image

The information stored for each blob includes maximum and minimum X and Y coordinate values, used later for centroid calculation, and the area of the blob. As new rows of pixels are added to a blob, the maximum and minimum X and Y values may be overwritten if the new row exceeds the current bounds of the blob. Additionally, the number of pixels in a new row is added to the existing blobs area. Because this method moves row by row, two growing blobs may converge at a later row. In this case, the data from one blob is transferred to the other blob, summing the areas, and updating the maximum and minimum X and Y values. The transferred blob is then nullified to prevent it from being considered in later calculations.

When all the blobs have been formed, they are then filtered by area so that just the largest blobs remain. These remaining blobs are then sorted so that only the largest 3 blobs are used. Because many of the detected blobs are small unwanted blobs due to noise in the image and unwanted red objects in the image background, the largest three blobs are most likely the red dots. If the blob exceeds the area threshold, the centroid is then calculated by finding the X and Y averages of the maximum and minimum values of the blobs bounds.

Position and Orientation Calculation

The next step is to determine which blob corresponds to which dot in the layout. For instance which blob is the top dot, which is the bottom right dot, and which is the bottom left dot? Due to the limitations of movement of the human head and the layout of the dots on the hat, the top dot, when visible to the camera, will always be higher than the other two dots no matter how the head is oriented. This allows for the top dot to be identified as the blob with the highest Y centroid component. Next, by no longer considering the blob identified as the top dot, the bottom right and bottom left dot can be identified as the blobs with, respectively, the highest X centroid component and the lowest X centroid component.

With each dot identified, their 2D centroid can be used to calculate a 3D position and orientation. For the X and Y position, the X and Y components of the centroid of the top dot are used.

$$X = \text{Top Dot } X \quad (1)$$

$$Y = \text{Top Dot } Y \quad (2)$$

The Z position is determined by the area of the top dot. One consideration with this relation is that if a flat circular dot is

used for the top dot, then the area will decrease as the head is turned to the side or up and down. To solve this, a spherical dot is used for the top dot so that it maintains the same area from any viewpoint. Instead of using the number of pixels in the top dot blob, the area is calculated by using the horizontal distance from the top dot's center x point to the maximum x point in the blob as a radius, and then finding the area using the formula $Circular\ Area = \pi r^2$. This method reduces the effect of lighting conditions on the z position calculation. If there is poor lighting, the edges of the spherical dot may appear as a different shade of red to the camera because of the way light is reflected off the sphere. Because of this, the edges might not be recognized and will reduce the number of pixels in the blob, thus distorting the Z-position. Additionally, the distance from the top dot's X center to the maximum X position in the blob is used as the radius to calculate the area because, when the user looks up, the camera's view of the top dot may become obstructed by the hat's brim, shortening the vertical axis and shrinking the dot's area significantly, yet it leaves the horizontal axis, and extracted radius, mostly undistorted, providing for a consistent method of calculation.

$$Z = \pi * \text{Top Dot Radius}^2 \quad (3)$$

The three components of the orientation, pitch, yaw, and roll, are more challenging to determine. Figure 6 graphically shows what measurements are used for pitch and yaw calculations in a sample case. For use in these pitch and yaw calculations, the X and Y components of the midpoint between the two front dots are determined by separately averaging the X components and Y components. The pitch is then determined by subtracting the Y component of the midpoint between the front dots from the Y component of the top dot.

$$\text{Pitch} = \text{Top Dot } Y - \text{Front Midpoint } Y \quad (4)$$

The yaw is similarly calculated by subtracting the X component of the midpoint between the front dots from the X component of the top dot. This is then divided by the distance between the two front dots to eliminate variance in caused by differing Z positions.

$$\text{Yaw} = \frac{\text{Top Dot } X - \text{Front Midpoint } X}{\text{Front Distance}} \quad (5)$$

These raw calculations of pitch and yaw are based on the camera resolution and the layout of the dots on the hat. To transform these to usable numbers, the pitch and yaw are multiplied by respective scaling factors to convert the raw numbers into angles. These scaling factors were determined by running the head-tracking software and measuring the values used in the pitch and yaw calculations as their maximum and minimum extremes, or when the head is turned to its practical limit. By associating the real angle of the head with these measured values, a linear interpolation is achieved for all values between the maximum and minimum angles. Although this is a linear interpolation for a non-linear 3D to 2D projection, the majority of the head movement will be within a range where a linear approximation is sufficient to achieve a very convincing effect. These values were calculated based on a red dot layout of particular dimensions. The front 2 dots are mounted 5 inches apart. The top dot is recessed 5 inches behind the front dots and 5 inches above the front dots.

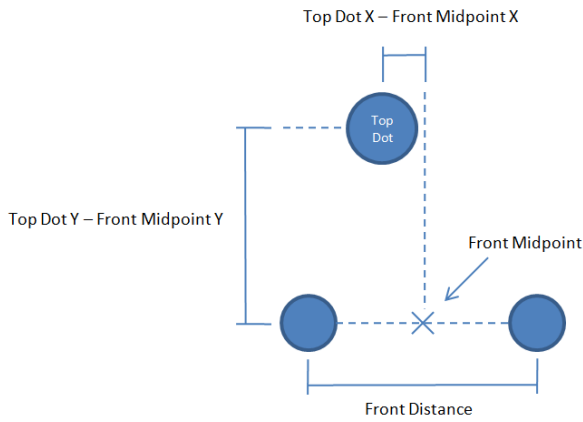


Figure 6: Pitch and Yaw Calculation Measurements

Finally, the roll is calculated by finding the angle the two bottom dots form with the horizontal. This is done by finding the inverse tangent of the difference of the Y components divided by the difference of the X components. This results in an angle in radians that is then converted to degrees. Figure 7 graphically shows this angle in a sample case.

$$Roll = \tan^{-1} \left(\frac{Right\ Dot\ Y - Left\ Dot\ Y}{Right\ Dot\ X - Left\ Dot\ X} \right) \quad (6)$$

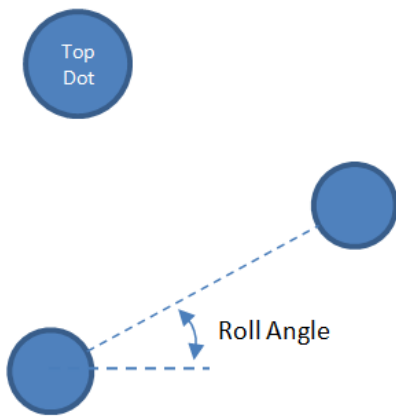


Figure 7: Roll Calculation

With these 6 position and orientation components calculated, they can now be communicated externally. Using a method of inter-process communication within the operating system, the values can be retrieved by a game or simulation process to be used to adjust the rendering of its 3D environment.

6. RESULTS

A fully functioning head tracking software solution was implemented. For development and debugging purposes, the head tracking software outputted the values of the calculated position and orientation on screen, which allowed for the programs behavior to be viewed and for the code to be fine tuned. In order to see the intended purpose of the software however, these values needed to be sent to a 3D game or simulation.

Once the head tracking software was programmed to communicate this data over a named pipe, provided by the Windows platform, a 3D program needed to read in these values. To demonstrate the full system from image acquisition to output 3D viewpoint adjustment, a demonstration 3D program was developed. This demo program is of a virtual camera located in 3D space. This camera is pointed at a three dimensional model of the words "TCNJ Engineering" to the front, and "Computer & Electrical" to the side. By wearing the 3 red dot layout and utilizing the head tracking software, the user can move his or her head around to look around the letters of "TCNJ Engineering" and see how the perspective changes based on their actual viewpoint. Additionally, by turning his or her head side to side, up and down, and tilting either way, the user can change the direction the virtual camera is pointed, and the words "Computer & Electrical", to the side, can be viewed.

The software successfully performs all the intended functions. Screenshots of the head tracking software working together with the demonstration program are provided with descriptions of what the images demonstrate. These images show the user and the head movements that correspond to the onscreen results. All six values, x-position, y-position, z-position, pitch, yaw, and roll are demonstrated.

The Normal Sitting Position is the view the user sees when in a natural sitting position. No viewpoint adjustment is applied. The user has a three position offsets: x, y, and z. By pressing the F11 key, the current position of the user is stored as his or her normal sitting position, about which final position values are calculated.



Figure 8: Normal Sitting Position

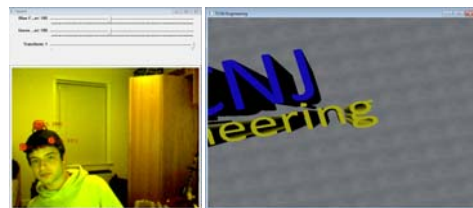


Figure 9: Leaning to Right (X-Position)

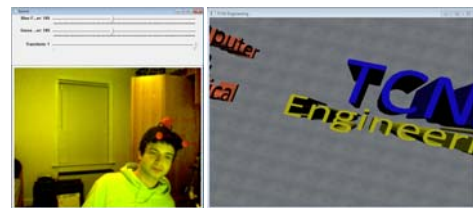


Figure 10: Leaning to Left (X-Position)



Figure 11: Leaning Forward (Y-Position)

Figure 11: Sitting Up in Chair (Y-Position)

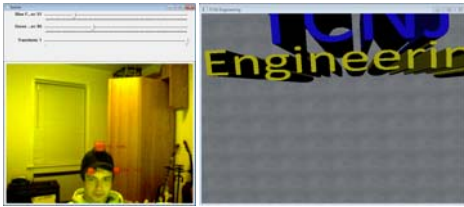


Figure 12: Sitting Down in Chair (Y-Position)

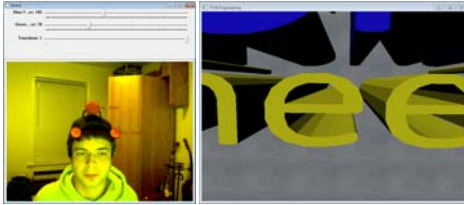


Figure 13: Leaning in Towards Monitor (Z-Position)



Figure 14: Leaning Away from Monitor (Z-Position)



Figure 15: Looking Up (Pitch)



Figure 16: Looking Up (Pitch)

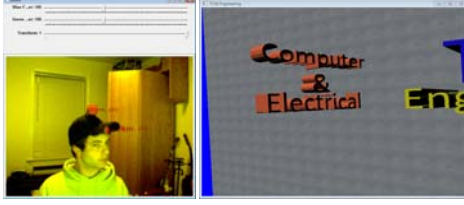


Figure 17: Looking Left (Yaw)

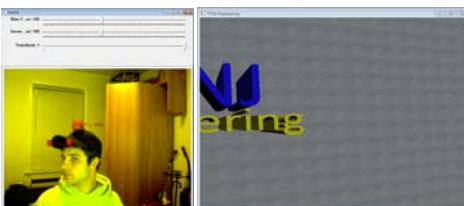


Figure 18: Looking Right (Yaw)



Figure 19: Tilting to Right (Pitch)

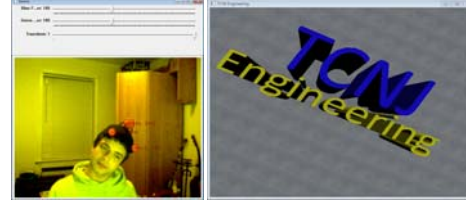


Figure 20: Tilting to Left (Pitch)

7. CONCLUSIONS

The head tracking system presented provides an inexpensive way to experience virtual reality on a PC. The end result when combined with a 3D application is an enhanced illusion of depth that works symbiotically with the 3D rendering. Also, the user has the ability to physically turn their head to look around the game world. With some modifications to applications, this system can be used with many existing and future games and simulations.

A working prototype with complete functionality and a 3D demonstration program have been completed. The six position and orientation components are correctly calculated and communicated to the 3D demonstration to adjust the virtual camera and verify the head tracking software's operation. Additionally, integrated calibration functionality allows the user to manually adjust the parameters of the image processing in real time. This allows for the head tracking software to achieve acceptable results in a wide variety of lighting conditions.

The head tracking system succeeds in translating head position and orientation from a 2D image into 3D position and orientation components that, when combined with a 3D rendering, provide an extra level of immersion to the user. Further research and development in this approach are investigated.

8. REFERENCES

- [1] Rulic, P.; Kramberger, I.; "Six degrees of freedom stereovision inside-out tracking," **EUROCON 2003**. Computer as a Tool. The IEEE Region 8, vol.2, no., pp. 52- 56 vol.2, 22-24 Sept. 2003.
- [2] Kruger, H.; Klingbeil, L.; Kraft, E.; Hamburger, R.; "BlueTrak - a wireless six degrees of freedom motion tracking system," **Mixed and Augmented Reality**, 2003. Proceedings. The Second IEEE and ACM International Symposium, vol., no., pp. 311- 312, 7-10 Oct. 2003.
- [3] Fung, J.; Tang, F.; Mann, S.; "Mediated reality using computer graphics hardware for computer vision," **Wearable Computers**, 2002. (ISWC 2002). Proceedings. Sixth International Symposium, vol., no., pp. 83- 89, 2002.
- [4] Liping Lin; Pingdong Wu; Jie Huang; Jian Li; "Precise Depth Perception in Projective Stereoscopic Display," **Young Computer Scientists**, 2008. ICYCS 2008. The 9th International Conference, vol., no., pp.831-836, 18-21 Nov. 2008.