Real-Time 3D Hand Tracking for 3D Modelling Applications

Roy Sirui Yang, Anthony Lau, Yuk Hin Chan, Patrice Delmas and Christof Lutteroth
Department of Computer Science
The University of Auckland, New Zealand
Emails: {syan095, alau023, ycha171}@aucklanduni.ac.nz, {p.delmas, lutteroth}@cs.auckland.ac.nz

***Abstract*—Problems arise when a user tries to operate in 3D spaces with 2D input devices. In search of an alternative type of Human Computer Interaction (HCI) in 3D modelling software, we present a working marker based stereo hand tracking prototype. Our system uses a combination of CAMShift and SSD window based stereo matching algorithm to achieve 3D tracking in real time, and then communicate with 3D modelling program Blender via a TCP port. This paper presents a detailed description of the system, some experiments we conducted to test for its performance, and discussions of current and future capability of the prototype.**

I. Introduction

3D modelling, especially digital sculpting, are widely used to build highly detailed and realistic digital props and characters for entertainment industries. While impressive, these models require a lot of times and efforts to create. Although the process of 3D modelling is similar to traditional props making and clay sculpting, the actual operations are often limited by 2D inputs from computer mice. While pressure-sensitive graphics tablets can improve the modelling experience, it is still impossible for 3D modellers to use their normal hand actions to directly manipulate 3D models. A natural human-computer interaction (HCI) interface that allows users to use their real world actions to control and to interact with computer programs for 3D modelling is desired.

At present, there are limited commercial solutions to assist 3D modelling. 3D mice from 3Dconnexion provide physical six degrees of freedoms to navigate a 3D viewport [1], but still users can only perform 3D modelling by using 2D mice. Sensable Technology produces a series of haptic-based devices that translate real world sculpting actions into digital sculpting [2], [3]. However, these devices tend to be expensive. The search for better and more cost-effective ways to create natural HCI for 3D modelling is a active field of research [4]–[9].

Some research aimed to maximise direct 3D mesh manipulation experience by means of semi-immersive virtual reality displays [4], [5]. With these prototypes, users are required to wear special spectacles and hold hand-held pointing gadgets to perform 3D operations. These extra gears might restrict natural body movements of a user during interactions with the systems.

Other research aimed for device-free approach. The idea is to let users interact with 3D programs with their natural body movements without any physical restriction implied from portable or wearable gadgets. Computer vision allow users to communicate with computers by tracking the user's movements, poses and gestures. Since computer vision was shown as possible device-free HCI interfaces for other applications [10]–[12], it was possible to apply similar technologies to create a device-free natural HCI interface for 3D modelling.

Kim et al. [6], [7] introduced a 3D modelling system that allowed users to use hand movements to do 3D modelling and to use simple hand gestures for further 3D mesh manipulations. It was operated by a stereo tracking system using ultra-violet light projections. This system used colour segmentation and stereo matching with Kalman filtering to perform their marker tracking [6]. Wang et al. [8] presented a real-time hand tracking system that tracked a colour glove worn by the user under normal room lighting. This prototype operated with a single commercial webcam. Motions, poses and gestures of the glove were tracked by the nearest-neighbour approach over a pose image database with a fast-searching data structure [8]. The system demonstrated its potential applications in animating 3D character, interacting with a 3D desktop or inputting text by sign-languages. The Sensors and Devices Group of Microsoft had recently created a hologram-like 3D interface that could fully interact with bare-hands of the user using their Kinect system, a single webcam and in-house tracking algorithms [9]. Apple [13] also acquired a new patent for developing 3D gesture recognition on future iPad device. They aim to use this new control interface to operate Computer Aided Design software on their future iPad. All these demonstrated that computer vision plays an essential role in providing a natural HCI for 3D modelling in general.

This paper presents a novel and working prototype that used marker-based stereo tracking to operate the 3D modelling program Blender in real time. The system utilised a low cost commercial stereo webcam to perform tracking. Real-time hand tracking is achieved by adopting the Continuous Adaptive Mean Shift (CAMShift) tracking algorithm and the Sum of Squared Difference (SSD) window-based stereo matching algorithm. Via a local Transmission Control Protocol (TCP) connection and a customised Blender plug-in script, the tracked data could be used to perform 3D viewport navigation in real time.

The paper first presents the overall system framework in Section II, followed by a detailed description of the Video Capturing Unit in Section III; 3D Tracking Unit in Section IV;

and Blender as a 3D Application Unit in Section V. The paper concludes with Section VI with remarks on our future research.

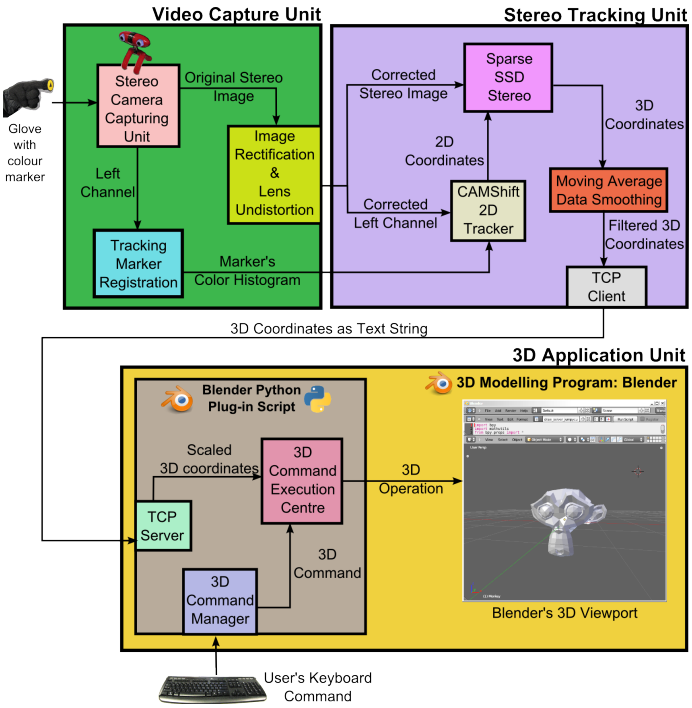

Fig. 1. Overall design of the current prototype

II. System Overview

Our research goal was to develop a cost-effective stereo hand tracking system that could be served as an alternative input device for an existing 3D modelling program. For the flexibility of changing hardwares and testing different tracking approaches in the future, we used a simple modular framework consisting of three main parts: Video Capturing Unit, Stereo Tracking Unit and 3D Application Unit (Fig. 1).

A. Video Capturing Unit

The Video Capturing Unit provides a video stream in real-time for the system. As our system uses stereo vision for tracking in 3D, the video unit will provide a synchronised stereo video stream for use in stereo matching. Generally, the chosen camera (or camera pair) needs to be calibrated, so that the camera parameters can be used to correct the image for lens distortions and alignment.

B. Stereo Tracking Unit

The role of the Stereo Tracking Unit is to apply stereo tracking techniques to obtain the 3D coordinates of the tracked target in real time. Support for hand gesture and pose recognitions will be a future extension for this unit.

Since the 3D Application Unit may reside in an external 3D modelling program, the stereo tracking unit should maintain an external communication interface for transmitting the tracking results and the extra command signals detected from the recognised poses/gestures.

C. 3D Application Unit

We chose to use existing 3D programs, rather than developing our own 3D demonstration platform from scratch. This is because existing 3D programs can provide us with access to reliable and ready-to-use 3D operations with supporting customization via run time script execution. This allows us to concentrate our efforts on developing and tweaking the overall hand tracking system.

For better internal access, the 3D application unit should be integrated into a 3D modelling program. One way for the integration is to develop the unit as a plug-in script under the internal scripting libraries that are provided by the selected 3D modelling program.

Apart from establishing a communication interface between the external Stereo Tracking Unit and the 3D program, the 3D application unit also processes the incoming 3D coordinates, identifies and executes the specific 3D operation requested by the user. The 3D command request can be signalled by keyboard commands, and in the future, hand signals given by the user, and recognized by the Stereo Tracking Unit.

III. Video Capturing Unit

Our system uses the Minoru 3D webcam, which consists of two VGA 640×480 CMOS sensors, and is capable of operating at 30 frames per second. The webcam is low-cost and communicates with a computer using standard USB 2.0 connections. Due to poor manufacturing quality, each webcam have slightly different camera parameters and alignment, hence the webcam was first calibrated by Zhang's method [14]. We have determined that the baseline is 60.7mm. The field of view in the horizontal direction is $41.94°$, and vertically $32.10°$. The angle of rotation between the 2 cameras is $0.537°$. The radial distortion coefficients of the left camera are ($\kappa_1 = -0.06876$, $\kappa_2 = -0.37504$), and the right camera are ($\kappa_1 = -0.06825$, $\kappa_2 = -0.31197$). By using the information from the calibration, the theoretical depth resolution for our stereo camera setup can be calculated (Fig. 2).

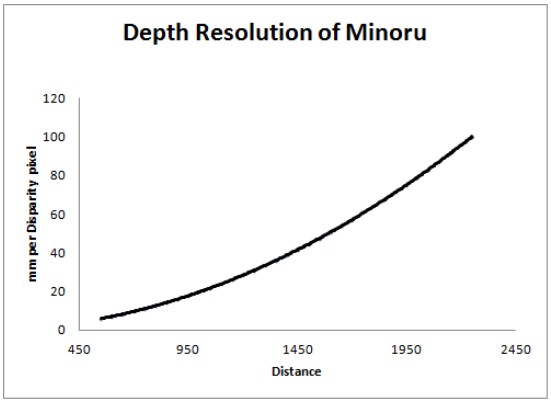

Fig. 2. Theoretical depth resolution of the Minoru Camera at different distances.

At the distance of 2000mm, movements less than 83mm can not be distinguished by the camera. This limits the effective

range of the entire system. After some testing we found the optimal distance for good performance in our system is 600mm to 1500mm.

IV. Stereo Tracking Unit

The Stereo Tracking Unit has three overall goals: 1. To track X and Y coordinates of the object (2D tracking), 2. Obtain Z coordinate and real world distance, and 3. Data smoothing and communication with the 3D Application Unit. The first goal can be done by using the basic CAMShift Tracking algorithm; the second goal was achieved by using sparse SSD window based Stereo Matching algorithm; The third goal was accomplished via a Moving Average filter to obtain a more stable 3D coordinates and a local TCP port to transmit the tracking results to the 3D Application Unit.

After a detailed description of the tracking unit, we present the results of some experiments we have conducted to test the performance of the tracking system.

A. Continuous Adaptive Mean Shift Tracking

CAMShift is a basic tracking algorithm introduced by Bradski in 1998, where he proposed a system that responded to the movement of the user by tracking the face of the user [15]. It was derived from Mean Shift [16], with additional abilities to update its search window dimensions and to estimate the size and 2D rotation of the tracked target.

Before the images can be processed by CAMShift tracking algorithm, first they need to be converted into a probability map, where the value of each pixel represents the likelihood of it being part of the tracked object. Similar to the approach taken by [15], our system constructs a 2D histogram of both the Hue and Saturation channel of the target, then treat the histogram as the probability distribution of the target object. Due to the fact that Hue becomes unstable at very dark and very bright pixels [15], our system ignores these pixels and treat the pixel as having probability of 0.

After the raw image was transformed into a probability map(Fig. 3a), The CAMShift algorithm is then performed on the probability map to determine the X and Y coordinates of the centre location of the tracked object, then estimate the object's width, length and orientation (Fig. 3b).

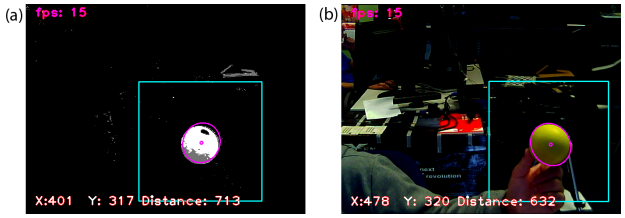

Fig. 3. User Interfaces of the Stereo Tracking Unit: (a) Probability distribution view; (b) Normal view. Despite using stereo camera, we only displayed video from the left channel. Cyan boxes & magenta ellipses indicated the current search window's position, orientation & dimensions on screen. The tracking result was shown at the screen bottom.

B. Sum of Squared Differences Stereo Matching

To achieve 3D tracking, the system needs to calculate the distance between the object from the cameras. By using stereo cameras, this can be achieved by the use of stereo matching algorithms.

We used the SSD window-based stereo matching algorithm [17], [18]. This algorithm takes a pixel on the left image, and finds the best match of a pixel on the right image, then uses the difference in their positions as the disparity level.

Considering the computation complexity [18] of the algorithm and performance of our system, only the disparity of the tracked object is calculated. Since the 2D position and size of the tracked object is already known from the previous CAMShift tracker, we can sample only a small number of points from the tracked object to be used to estimate the depths of the object.

By assuming the tracked object has a relatively flat surface, extreme disparity levels are considered to be errors. To remove extreme values efficiently the system will rank and discard the top and bottom 20% of the disparity values from all the sample points, then take the average of the rest. As extreme outliers are removed, this increases the likelihood of obtaining a more reliable depth estimate.

To convert the disparity values obtained from the stereo matching algorithm into a measure of real world distance Z, the following formula (Eqn. 1) is used:

$$Z_w = \frac{f \times b}{d} \tag{1}$$

Where Z_w is the real world distance, f is the focal length of the camera, b is the baseline between stereo cameras and d is the disparity.

C. Smoothing of coordinates and TCP connection

Partly due to the fact that our system is working with low cost cameras, it was noticed that the large amount of error reduces the stability of the tracking result. To resolve this issue, the coordinates obtained from the tracker are smoothed using a moving average filter, using the average of the current and 5 previous frames as the final coordinate of the object.

From Eqn. 1, we see that as the distance (Z_w) increases, errors in the disparity translating to errors in the distance will be amplified. At a distance of 1000mm, the fluctuation due to error in Z_w measure cannot be smoothed by the moving average. An extra precaution is taken, where a threshold filter is applied to any changes in the distance. After some testing, an empirical value of 4% is chosen, meaning all changes in Z_w that are less than 4% is ignored. The tracker can achieve stable tracking in the distance measurement, while still sensitive to any intentional movement made by the user.

After the final 3D coordinates are computed, the coordinates are sent via a local TCP port to the 3D Application Unit. The using of a TCP port allows the Stereo Tracking unit to communicate with any program with a working TCP server. This increases the flexibility of the Stereo Tracking Unit, for it can easily be ported into any other programs.

D. Experiments and Performance of the 3D Tracker

We have conducted two experiments to evaluate the accuracy of the stereo tracking unit. In the first experiment, tracked object's real movements are compared with the movements detected by the tracker. We conducted several tests where a ball has been moved along a rail in a straight line while its coordinates are recorded by the tracking unit(see Fig. 4).

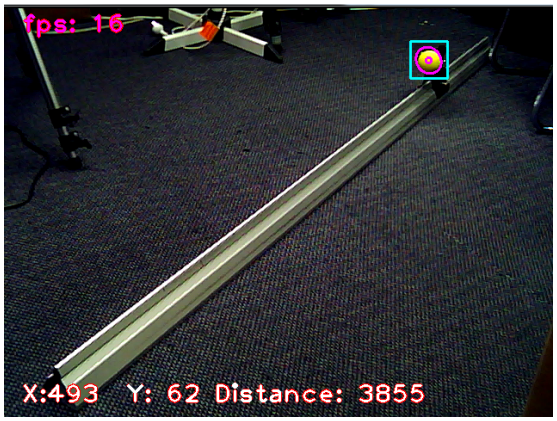

Fig. 4. A marker on the top of the rail was tracked to move in a straight line in the real world.

The coordinates obtained from the tracker are recorded and plotted in 3D (Fig. 5). As it is known that the path of the ball is a straight line, the accuracy of the tracking unit can be estimated based on how well the tracker's reported coordinates fit a line. The image X and Y coordinates are first translated into real world X and Y coordinates using equation:

$$(X_w, Y_w) = \frac{(X, Y) \times Z_w}{f} \tag{2}$$

where (X_w, Y_w) is the real world coordinates of a pixel, (X, Y) is the images coordinates of the pixel, and f is the focal length of the cameras.

Since the real world movement of the object is a straight line, with large enough measurements the line that best fit the measured coordinates is expected to be identical to the real world projectile of the object. Then how well the coordinates measured by the tracker fits the line of best fit reflects the accuracy of the tracker. To investigate further, a 3D line of best fit is computed for each test, and the closest distance of each data point to the line of best fit (residual) of all the data points are calculated.

In the second experiment, the actual distance of a tracked object is compared with the Z_w measured by the 3D tracking unit. As can be seen in Fig. 6, the two variables are highly correlated (R^2 = 0.9968). The trend line indicates the degree of correlation between measured distance and tracking results. It has a gradient of 1.0122, a value very close to 1. This means that the tracking unit can accurately track depth within the range between 600mm to 2000mm.

Test	mean of Residuals (mm)	Stand Deviation of residuals(mm)
1	2.1536	0.9758
2	10.9538	5.8524
3	17.4734	15.242
4	13.4444	6.9689
5	8.6004	7.1142
6	6.4940	3.6470
7	3.7392	3.4419

TABLE I
TABLE SHOWING THE MEAN AND STAND DEVIATION OF RESIDUALS OF THE COORDINATES MEASURED BY THE TRACKER AND THE LINE OF BEST FIT. ALL MEASUREMENTS ARE IN MM

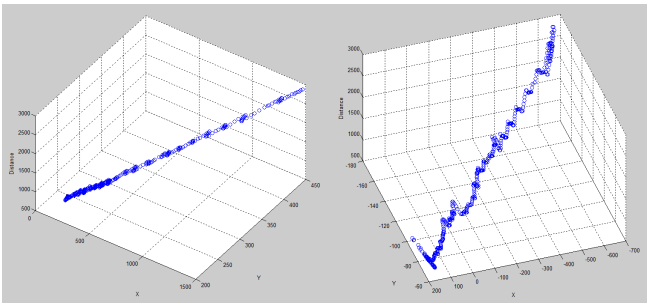

Fig. 5. 3D plots displaying the results from 2 configuration of rail placement. The data points are translated real world X, Y and Z coordinates.

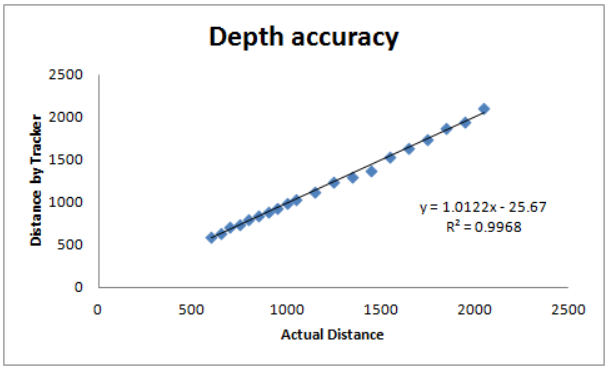

Fig. 6. The actual distances from the object to the camera compared with those obtained from the 3D Tracking Unit.

V. 3D Application Unit

Apart from being free and open source, Blender [19] was chosen because the program provided digital sculpting functions and supported Python as its internal scripting language. It had a dedicated Blender Python API, to access functions within Blender. The API granted programmers more powers and flexibilities in extending Blender. Our current 3D Application Unit was developed as a plug-in script for the official stable build (r39307) of Blender 2.59.

The current unit consisted of three components (as shown in Fig. 1): the Background TCP server, the 3D Command Manager and the 3D Command Execution Centre. The current unit was designed to work best with a single 3D viewport. Via the hand tracking system and keyboard controls, it allowed the user to perform basic 3D view navigations in the single viewport.

A. Background TCP Server

As Blender Python did not provide any external communication support, we decided to use standard Python's socket programming to create a local TCP server to create the necessary communication port for our external stereo tracking unit. This TCP server was designed to listen for any incoming traffics from its client, the Stereo Tracking Unit. Once received, it converts the text string received from TCP client into the corresponding 3D coordinates. After scaling and adjustments for Blender's 3D world, the stream of 3D coordinates are continuously supplied to the 3D Command Execution Centre for any specific 3D operation requested by the user.

To constantly monitor for any incoming packet through the assigned TCP port, our TCP Server needed to run concurrently with Blender. However, the current Blender Python API had limited support over threading [19]. This threading restriction was resolved by setting up the TCP server to run inside Blender's redraw callback.

By introducing an auto-resizing 3D cube into the 3D scene, the Blender's redraw callback routine was triggered regularly to renew the screen display. During each redraw routine, the TCP server checks for any incoming traffic from the local network. Since each redraw callback interval lasts for about 6 milliseconds (or 167 frames per second), this allows the TCP server to operate as if concurrently with Blender in the background.

B. 3D Command Manager

3D Command Manager was responsible for identifying 3D command requests and managing all internal signals within the 3D Application Unit. The 3D Command Execution Centre relied on these signals to determine how to process the steams of 3D coordinates for carrying out the requested 3D operation.

To simplify the current prototype development, we used a small number of keyboard keys to represent 3D commands available when Blender was operated by our hand tracking prototype. Thus, the current 3D Command Manager was capable of detecting keyboard key presses to identify 3D command request from the user. This design serves as a placeholder for the future extension of posture and gesture recognitions in the tracking unit.

In Blender 2.5X, tools/functions were introduced in the forms of operators [19]. Once the script was run, the operator was registered into Blender as an available function inside the Blender Toolbox menu. This menu could be called out anytime to select new functions in Blender. When the desired operator was selected, it would override Blender's normal operations temporarily until its specific tasks were accomplished. This behaviour allows us to define our special keyboard shortcuts for the hand tracking mode freely without conflicting the original Blender keyboard shortcuts.

C. 3D Command Execution Centre

Based on the received command signal, the 3D Command Execution Centre processes the input streams of 3D coordinates and applied the results into the execution of the desired 3D operation. At present, this component only provides 3D view navigations (i.e. view translation, rotation and zooming) within a single 3D viewport. These 3D operations were implemented in priority because 3D view navigations were essential tools for assisting 3D modelling. Although Blender could display multiple 3D viewports, our current tracking prototype were unable to detect which particular 3D viewport the user was trying to access during hand tracking. Thus, we limited our 3D operation supports to just a single 3D viewport.

The 3D viewport behaved like a 2D plane whose centre was always tangential to a virtual sphere in the 3D world. Parameters provided by Blender Python API for viewport navigations could be treated as the parameters of this virtual sphere.

Therefore, view translation was achieved by moving the sphere centre around (Fig. 7a&b). Since the orientation of the current viewport was measured with respect to the virtual sphere centre, viewport was rotated about this virtual centre(Fig. 7c&d). Adjusting the radius for the virtual sphere created the zooming effect at the 3D viewport (Fig. 7e&f).

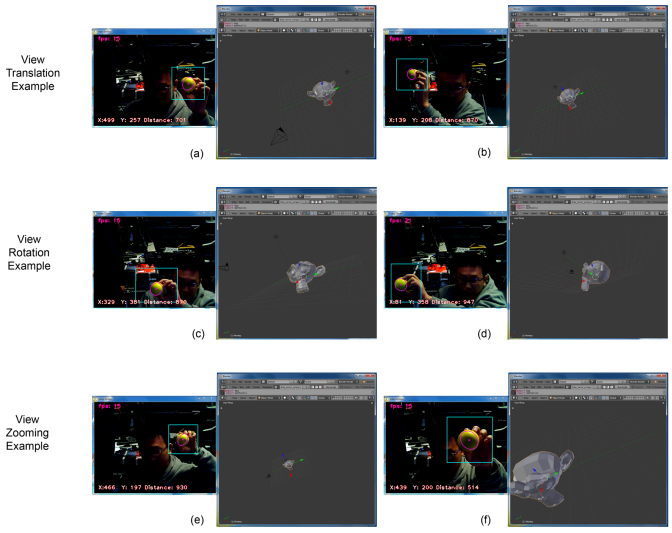

Fig. 7. Example snapshots of the current tracking prototype in action to navigate the 3D viewport in Blender

The magnitudes of 3D translations for the 3D viewport was obtained from the difference between the current and the starting positions of the tracked marker. The difference in view depths of the marker provided the change in view zooming.

The rotation of the viewport was calculated by the virtual trackball approach [20]. The 3D coordinates received in the current frame was assigned as the next starting coordinates after each evaluation. This enabled the viewport to quickly adapt any change in rotation axis due to sharp movement change from the marker, e.g. from horizontal to vertical movements.

VI. Conclusions and Future Works

In this paper, we have presented a prototype of our new system, which combines computer vision with a 3D modelling

software. By using 3D object tracking algorithms with a low-cost stereo webcam, our system achieved real time 3D tracking at approximately 15 frames per second. We also presented the results that test the performance of the tracker and demonstrated that our system can produce accurate tracking result within the optimal range of 600mm - 1500mm. Our system was able to communicate with Blender via a TCP connection and, by internal scripting, to gain access to a number of 3D operations. At the current stage, the combined system allows the user to navigate within a 3D space in Blender by simply hand movements.

As the current system is just a prototype, there are rooms for future improvements to the system. This future system should be able to track multiple objects. It should also be integrated with gesture recognition such that the user can switch between 3D commands by simple hand poses/gestures, rather than key pressing. 3D Application Unit will be introduced with more 3D operations, especially those related to mesh editing. We hope that the full system should give users a more pleasant experience in 3D modelling as they can explore many possibilities brought about by the ability to use their natural hands to directly manipulate the 3D objects.

REFERENCES

[1] "3Dconnexion," [Retreived on: 30 June 2011]. [Online]. Available: http://www.3dconnexion.com

[2] T. Massie, "A tangible goal for 3d modeling," *Computer Graphics and Applications, IEEE*, vol. 18, no. 3, pp. 62–65, 1998.

[3] "Sensable," [Retreived on: 30 June 2011]. [Online]. Available: http://www.sensable.com

[4] S. Schkolne, M. Pruett, and P. Schröder, "Surface drawing: creating organic 3d shapes with the hand and tangible tools," pp. 261–268, 2001.

[5] T. Harviainen, L. Svan, and T. Takala, "Usability testing of virtual reality aided design: Framework for prototype development and a test scenario," in *4th INTUITION International Conference on Virtual Reality and Virtual Environments*, 2007, pp. 172–181.

[6] H. Kim and D. W. Fellner, "Interaction with hand gesture for a back-projection wall," in *Computer Graphics International*, 2004, pp. 395–402.

[7] H. Kim, G. Albuquerque, S. Havemann, and D. Fellner, "Tangible 3d: Hand gesture interaction for immersive 3d modeling," in *Virtual Environments*. Eurographics, 2005, pp. 191–199.

[8] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–8, 2009.

[9] Microsoft Research Sensors and Devices Group, "HoloDesk – Direct 3D Interactions with a Situated See-Through Display," [Retreived on: 5 Nov 2011]. [Online]. Available: http://research.microsoft.com/apps/video/default.aspx?id=154571

[10] C. Manresa, J. Varona, R. Mas, and F. J. Perales, "Realtime hand tracking and gesture recognition for human-computer interaction," *Electronic Letters on Computer Vision and Image Analysis*, vol. 0, no. 0, pp. 1–7, 2000.

[11] T. Kurata, T. Okuma, M. Kourogi, and K. Sakaue, "The hand mouse: Gmm hand-color classification and mean shift tracking," in *IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, 2001, pp. 119–124.

[12] C. Shan, Y. Wei, T. Tan, and O. Frédéric, "Real time hand tracking by combining particle filtering and mean shift," in *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, 2004, pp. 669–674.

[13] Patently Apple, "Apple's Wild New 3D Gesturing is Aimed at CAD, Avatar Creation & More," [Retreived on: 5 Nov 2011]. [Online]. Available: http://www.patentlyapple.com/patently-apple/2011/07/apples-wild-new-3d-gesturing-is-aimed-at-cad-avatar-creation-more.html

[14] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[15] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *Fourth IEEE Workshop on Applications of Computer Vision*, 1998, pp. 214–219.

[16] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *Information Theory, IEEE Transactions on*, vol. 21, no. 1, pp. 32–40, 1975.

[17] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: theory and experiment," in *IEEE International Conference on Robotics and Automation*, vol. 2, Sacramento, Califomia, 1991, pp. 1088–1095.

[18] Wikipedia, "Template matching — Wikipedia, the free encyclopedia," 2011, [Retreived on: 13 Oct 2011]. [Online]. Available: http://en.wikipedia.org/wiki/Template_matching

[19] Blender Foundation, "blender.org," [Retreived on: 17 Oct 2011]. [Online]. Available: http://www.blender.org/

[20] E. Angel, "Interactive computer graphics: A top-down approach using opengl, section 4.10.2," 2003.