3D STYLUS: AN INTUITIVE 3D POINTER FOR VOLUMETRIC RADIOLOGICAL DATA

by

Enver Yagcı

B.S., in Electrical Engineering, Yıldız Technical University, 2003

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in FBE Program for which the Thesis is Submitted

Boğaziçi University

2007

3D STYLUS: AN INTUITIVE 3D POINTER FOR VOLUMETRIC RADIOLOGICAL DATA

APPROVED BY:

Assist. Prof. Burak Acar .....
(Thesis Supervisor)

Prof. Fikret Gürgen .....

Prof. Lale Akarun .....

DATE OF APPROVAL: 01.06.2007

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor Assist.Prof. Burak Acar for guidance and help during the preparation of this dissertation.

I also would like express my gratitude to Prof.Dr. Fikret Gürgen and Prof.Dr. Lale Akarun for the time they spend to examine my thesis and being in my thesis jury.

I'm also grateful to TUBITAK for the scholarship for my expenditures.

I also want to thank Ugur Bozkaya, Ph.D. Student in biomedical engineering for his assists.

I have very much benefit from the discussions with my dear friends in the VAVLAB, among whom I would like to mention especially Deniz Diktaş, Sıla Girgin, Erkin Tekeli and Erhan Durusüt.

I want to acknowledge my parents support and I would like to send my especial thanks to them.

I owe to Tuğba Evrim for her incomparable company. She was always with me.

ABSTRACT

3D STYLUS: AN INTUITIVE 3D POINTER FOR VOLUMETRIC RADIOLOGICAL DATA

This work introduces a new human-computer interface(HCI) device for three dimensional (3D) visualized data. Although there are different approaches developed in the literature, 3D interfaces have not finished their development.

The HCI we designed basically aims at taking an arbitrary cross-section (oblique slicing) from volumetric data. Additionally, the interface we called "3D Stylus" has an extra option which enables the user to draw two dimensional curves.

The *3D Stylus* is a colorful, long and thin structure like a pen. The *3D Stylus* is tracked by using image processing algorithms. 3D information about the interface is acquired with the help of stereo processing techniques applied to the images come from the stereo camera.

The *3D Stylus* is a flexible tool that can be used for different purposes in different applications. In the thesis, we firstly develop a oblique slicing application using *3D Stylus* as normal vector of slicer held by the user. Then another application is developed for 2D drawing, and the *3D Stylus* used as a chalk.

At the end of the thesis, experiments are presented. Users are asked to use the *3D Stylus* and mouse interfaces in our applications. Results about efficiency of the *3D Stylus* are given, compared with mouse.

ÖZET

3D STYLUS: HACİMSEL RADYOLOJİ VERİLERİ İÇİN SEZGİSEL 3B GÖSTERİCİ

Bu çalışma yeni bir insan-bilgisayar arayüzü ortaya koymaktadır. Her ne kadar literatürde yapılmış farklı çalısmalar olsa da, üç boyutlu(3B) ara yüzlerin gelişimi henüz tamamlanmamıştır.

Tasarladığımız arayüz temel olarak hacimsel bir veriden rastgele kesit almayı hedeflemektedir. Buna ek olarak, "3B Gösterici" adını verdiğimiz arayüzümüzün kullanıcıya iki boyutlu çizim yapma imkanı veren bir seçeneği de mevcuttur.

3B Gösterici renkli, uzun ve ince bir yapıda olup kaleme benzemektedir. 3B Gösterici görüntü işleme algoritmaları kullanılarak takip edilmiştir. Göstericiye ait 3B bilgisi ikili işleme tekniklerinin yardımıyla ikili kameradan alınan resimlerden elde edilmiştir.

3B Gösterici farklı uygulamalarda farklı amaçlar için kullanılabilecek esnek bir yapıya sahiptir. Bu tezde önce 3B Göstericiyi, kullanıcının tuttuğu kesit alan yüzeyinin normal vektörü olarak kullandık. Ardından iki boyutta çizim için bir uygulama geliştirip, 3B Göstericiyi bir tebeşir gibi kullandık.

Tezin sonunda, deneylere yer verilmiştir. Kullanıcıdan uygulamalarımız için arayüz olarak mouse ve 3B Göstericiyi kullanmaları istenmiş ve 3B Göstericinin verimliliği hakkındaki sonuçlar fare ile karşılaştırmalı olarak verilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

CTC	Computed Tomography Colonography
MRI	Magnetic Resonant Imaging
MD	Medical doctor
fMRI	functional Magnetic Resonant Imaging
OpenCV	Open computer Vision Library
VTK	Visualization Toolkit
IPL	Inter Image Processing Library
HCI	Human Computer Interface
CPU	Central Processing Unit
DOF	Degree of Freedom
PCA	Principal Component Analysis
LS	Least Squares
EKF	Extended Kalman Filter
UKF	Unscented Kalman filter

1. INTRODUCTION

There have been significant advances in medical image processing in last two decades. Numerous studies have been carried out for locating interesting regions or structures in human body. Different type of medical modalities such as CTC, MRI, PET are producing higher resolution images compare to even one year ago. As an insight about resolution of these modalities, we can tell that a CTC could produce a volumetric image up to 2048^3 voxel. It is clear that as the data is getting bigger, medical doctors' (MDs) information about patients and diagnosis of illness getting more efficient and accurate than before. On the other hand investigating all huge amount of data is getting more difficult than investigating in the past. As long as MDs benefit from higher resolution image, it is meaningful to develop higher resolution modalities. For convenience, most of the time, MDs prefer to see these images with the help of advanced visualization techniques. Thanks to state-of-the-art visualization techniques, displays can show data that we are interested in as a volume at a time on the screen. However volumetric images on the screen are not suitable for MDs aim. Any volumetric data needs to be sliced to understand what is happening inside the data. There are some proposed techniques or tools to help people to see both volumetric data and its slice view. In some cases MDs face with the difficulty of using due to weakness of HCI or limitation of slicing. Currently researches are still after a design works intuitively and at the speed of user.

To sum up, All rapid developments of the radiological devices and visualization technologies with growing available 3 dimensional (3D) data are demanding new human computer interfaces (HCI). HCI need especially emerges from difficulties that are seen during the investigation of patient information from radiological devices. In this thesis, we will try to design a HCI using computer vision methods and evaluate its efficiency.

2. BACKGROUND ON HUMAN COMPUTER INTERFACES FOR 3D DATA

In today's computer world, Almost every user always faces with "mouse and keyboard". There is no doubt that people are accustomed with "mouse and keyboard" and they have nearly completed their evolution. Although "mouse and keyboard" is designed for 2D application before 80s, any development in computer world in terms of CPU speed or software complexity cannot substitute any different HCI. The fact that "mouse and keyboard" is not a convenient tool for manipulating visualized 3D data on the screen is not a sufficient reason to change human computer interfaces from mouse to another interface alone. Whenever you need to work with 3D data, programs need a mechanism that switches dimension otherwise, during the usage an ambiguity will be emerged for user. The ambiguity is to understand which mouse motion corresponds to which dimension pair. Because there are 3 dimension virtually on the screen and mouse motion has 2 axis on the surface, the plane which mouse live could correspond one of three pair (x-y, y-z, x-z). it is obvious that confusing with the possible axis pairs is very likely. Generally this problem could be solved by a switching mechanism between pairs or by waiting that user could solve axis from the clues on the screen depending on the position of mouse. In fact current programs generally prefer the later, so programs work with ambiguity instead of changing dimension for mouse. On the other hand, this ambiguity could be more preferable than using some other HCIs. The reason to choose mouse instead of other interface can originate from different reasons, like reluctancy to new design, difficulty of interpretation, price of HCI, etc.

At this point, we should also emphasize that current display technology is not sufficient for 3D visualization. It is not always possible to determine the 3D object orientation by viewing its projection on a screen. This confusion of display is very similar to that of mouse. Therefore you may interpret a plane lies along x-y axis as if it is along y-z axis and so on. Especially, is plane symmetric, users suffer this problem more. However, as HCIs are developed, new display technologies will be developed by

researchers to visualize 3D data.

2.1. Literature Review on 3D HCI

There are many different approaches to develop an ergonomic HCI for 3D interactions [1–3]. Mainly, there are two main approaches in the literature namely, computer vision and image processing based techniques [2] and electronic sensors and their output signal based methods [1].

Main advantage of second approach is its efficiency in terms of resolution compared to first one. Contrarily, its high cost due to sensor prices effect negatively. Considering that 3D HCI won't be as invasive as mouse-keyboard pair, it doesn't seem like prices go down dramatically like some other computer peripherals.

On the other hand, the main advantage of camera based system is necessity of camera in different application areas from industry to academic works. This wide utilization leads reduction of their prices dramatically nearly every year. Another advantage, there is no mechanical constraint that is valid for many sensor based applications. Nevertheless it is too difficult to make computer vision based system as robust as the sensor based system, because of number of pixels in the area or vulnerability of different noise like changes of lighting source. Some good examples of these two approaches will be covered in the below.

The *CAT* [1] is one of the good examples of solution for sensor based approaches. The *CAT* is a 6 degrees of freedom (DOF) input device looking like a classical circular table. The table-top can be oriented in space thanks to three nested rotation axes called Yaw, Pitch, and Roll. Photo of the *CAT* is given in the Figure 2.1[1] . It has two different modes, one for 3D and second mode is 2D. In 3D mode, the basic functionality is manipulating virtual objects in space or controlling camera viewpoint trajectories. During the manipulating orientation of virtual objects; the orientation of the tabletop mapped to the orientation the 3D scene. Both table top and the scene always have

[1] iparla.labri.fr/publications/2003/HGRT03/cat.pdf

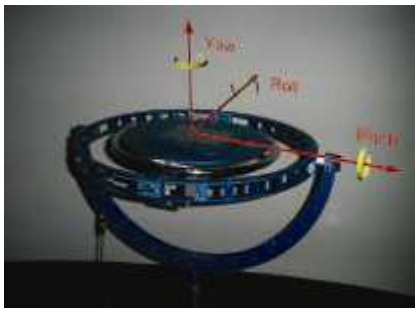

Figure 2.1. Picture of CAT when is being used. Rotation axes are also illustrated with red arrows

the same orientation. When controlling camera view point, the rotation of the camera viewpoint is directly mapped onto the rotations applied to the table top.

About 2D interaction, the table top of the *CAT* represents a physical workspace which can be used for 2D interaction. The table fixed on the *CAT* top precisely recovers the position of a pen on it.

The mechanical construction of the *CAT* provides a direct feedback concerning rotations. Although any rotation is theoretically possible with the *CAT*, some few of them are hard to perform because of the nested axes rotation. *CAT* is basically suitable for some virtual reality applications and manipulating data in front of large screen.

The second device that we can mention as a sensor based design is based on hand tracking, called *CyberGlove* [2]. *CyberGlove* is also commercially available. *CyberGlove* can be described as a fully instrumented glove that provides up to 22 high-accuracy joint-angle measurements. The 22-sensor model features three bend sensors on each finger, four abduction sensors, plus sensors measuring thumb crossover, palm arch, wrist flexion, and wrist abduction. It has wireless communication module therefore, one of the common problems of sensor based systems (cable restriction)is overwhelmed.

Figure 2.2. CyberGlove is seen on hand of user with its wireless module on the wrist

It uses proprietary resistive bend-sensing technology to accurately transform hand and finger motions into real-time digital joint-angle data. This system is especially developed for virtual driver tests. Figure 2.2[2] shows *CyberGlove* during the test with driving software.

The *CyberGlove* features Virtual Technologies' patented resistive bend-sensing technology that is linear and robust.

A good example of the camera based systems is proposed by [3]. A user has interaction devices in his hands, like a marked pen, cube or thimble. Using infrared lighting, two cameras with infrared-pass filters, and retro-reective markers are tracked. These interaction devices can easily be constructed and applied. Device recognition is based on projection invariant pattern characteristics. Reconstruction is realized through stereo correspondence and so-called the epipolar geometry. Once the correspondence between left and right camera images is achieved and knowing the internal and external camera parameters, the position of the marker in the 3D environment can easily be determined.

Another work [4] uses a different means of visualization as well as interaction. This system is a package with a visualization and an interaction part. They use a 3D volumetric display which generates a 10" spherical 3D volumetric image by sweeping

[2]http://www.immersion.com/3d/products/cyber_ glove.php

198 2D image planes around the vertical axis.

Some markers which is very bright are developed to use for interaction. A motion tracking system issued to track the 3D positions of these markers placed on the user's fingers. System uses several high-resolution cameras to track the 3D location of multiple passive reflective markers in real time. In addition to tracking the location of the markers in 3D space, the system can uniquely identify and label each marker according to its position on a users fingers. Application software was written in C++ and OpenGL, with a custom OpenGL driver specific to the volumetric display. Similar to interfaces for 2D touch screens, they display frequently used commands as buttons on the surface of the display. They also uses posture of the hand, they developed a set of hand postures and gestures which can be carried out on or off the surface of the display. The Surface Browser displays various objects by organizing them into cells of a 2D array. Four such arrays, or pages, are then projected around the entire inner surface of the display, allowing the user to easily interact with the objects by touching the surface of the enclosure directly above them. While performing any of the transformations, a colored 3D icon is drawn at the center of the model, indicating which transformation is currently being applied. They combine this data with the precise topology and 3D spatial location of the display's enclosure to simulate an enhanced touch sensitive display. Figure 2.3[3] shows the display and interaction system.

Fleisch et al. [5] has proposed more than one interaction tools in order to satisfy different needs. System ,called ART by Fleisch, consists of cameras and tools for interaction. All tools are based on light tracking. To track the user's head and hand positions they use an optical tracking system from ART, which provides high accuracy and low latency. A redundant amount of cameras and markers is used to overcome the disadvantage of keeping the camera's line of sight clear. In addition optical tracking allows for untethered objects, such as a wireless pen. While moving the 3D pen device in space the tracker continuously delivers position events. These positions are used as sample points. The shape of the created NURBS curve [6] depends on the distances between the sample points. The result is therefore always a trade off between com-

[3]http://www.dgp.toronto.edu/papers/tgrossman_UIST2004.pdf

Figure 2.3. A user is seen above while using especially designed display. He interact with display with some markers on his hand

plexity of the curve and the distances between sample points. It detects points of high curvature and cuts the curve at these points. The resulting curve then consists of partial curves with few control points and possible sharp edges at the junctions. Figure 2.4[4] and Figure 2.5 illustrates system and available interactors respectively

Another image processing based system which tracks hand is shown in Figure [7]. The user interface is used in a framework developed. The name of the framework is MASTER-PIECE and it integrates gesture and speech modalities into a designer and assembly application so as to increase the immersion of the user and to provide a physical interface and easier tools for design, than the mouse and the keyboard. Moreover, the user is capable of generating simple 3D objects and search for similar 3D content in a database, which is nowadays another very challenging research topic. Head and hands are detected and tracked. Their positions in the 3D space are given by a stereo camera. The face of the user is automatically detected by an artificial neural networks. The hands are detected as skin colored moving zone in front of a vertical plane passing through the head, triggering pointing gesture recognition. Once detected, the body parts (head and hands) are simultaneously tracked until tracking

[4]T.Fleisch, G.Brunetti, P.Santos, A.Stork, Stroke-Input Methods for Immersive Styling Environments, *Proceedings of the Shape Modeling International*, 2004

Figure 2.4. A snapshot while ART is being used. Every component has bright markers and especial aim.

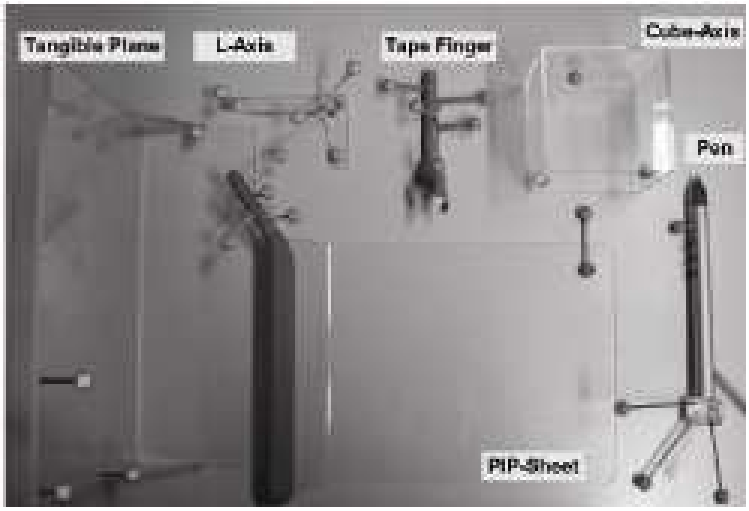

Figure 2.5. All available tools of ART can be seen above

failure is automatically detected. Then the detection for the lost part is re-triggered. The tracking process aims at explaining each new image by a statistical model with the EM algorithm. A screen-shot during the usage of the system is shown in Figure 2.6

Betke et al. proposed "camera mouse" [8] for people with severe disabilities. Camera mouse tracks user's movements with a video camera and translates them into the movements of the mouse pointer on the screen. The Camera Mouse system currently involves two computers that are linked together. They call these computers vision computer and user computer. A schematic plan of the system is shown in Figure 2.7. The functionalities of the two computers could be integrated into one computer, nevertheless their current setup assures sufficient processing power for the visual tracking and allows a supervisor to monitor the tracking performance without interrupting the users actions. The user or an attending care provider clicks with the vision computers mouse on the feature in the image to be tracked, perhaps the tip of the users nose. Therefore tracking point is depending on the selection in each tracking trial. The cameras remote control can be used to initially adjust the pan and tilt angles of the

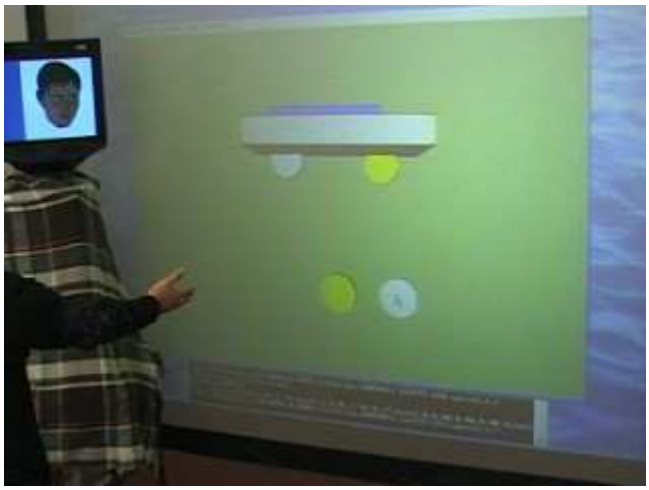

Figure 2.6. User is drawing a cylinder using hand gesture in front of white screen. The camera track the user is at the top of the screen

camera and its zoom so that the desired body feature is centered in the image. The vision system determines the coordinates of the selected feature in the initial image and then computes them automatically in subsequent images. System uses features such as the tips of the user's nose or finger during the test phase. A dozen people who cannot speak and have very limited voluntary muscle control had tried using the Camera Mouse to access the computer. Ten have cerebral palsy. Two have traumatic brain injury: one from an automobile accident and one from a motorcycle accident. After tests are over they asked them whether they want to use this design or not. According to their report, Nine of the people are continuing to use the Camera Mouse. Six can use the Camera Mouse to spell using an on screen keyboard system. Three of the people did not have sufficient muscle control to use the Camera Mouse. They are using Eagle Eyes to control the mouse pointer by moving their eyes. One of the people who was not successful was close and will be given additional opportunities to try the Camera Mouse in the future. it was shown by tests, that one of the disabled people could even play video games using the camera mouse. In their publication, they also reveal some good results on healthy people.

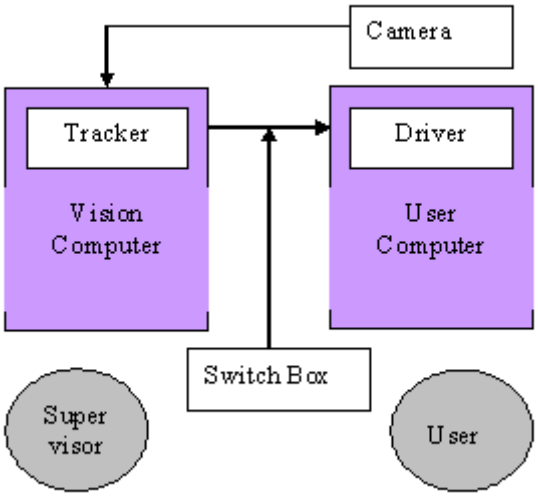

Figure 2.7. Camera mouse system components and their connectivity between other components are depicted above

Hermann T. et al. use hand and arm gestures without using any glove or markers [9]. Its central element is a gesture box containing two infrared cameras and a color camera which is positioned under a glass desk. Gesture box can be seen in Figure 2.8 and it is below the transparent desktop on the floor. Arm and hand motions are tracked in three dimensions. In their application , an interactive real-time browsing and querying of auditory self-organizing maps is realized. Moving the hand above the desk surface allows to select neurons on the map and to manipulate how they contribute to data sonification. Each neuron is associated with a prototype vector in high-dimensional space, so that a set of 2D-topologically ordered feature maps is queried simultaneously. Auditory maps [5] are visualized maps, that have attached to each location in representation space (map space) a high-dimensional data vector that is presented acoustically. However, parts of the map can also be presented visually in the map, e.g. by using spatially resolved color or texture.The level of detail is selected by hand altitude over the table surface, allowing to emphasize or deemphasize neurons on the map

All these works that we mentioned above implies that for 3D interfaces we need different solution for different problems. To find a unique solution for all applications, is a big challenge. there is consensus that HCI for interacting with 3D data/scenes must be application specific.

Figure 2.8. Model of the gesture desk, obtained by ray tracing of a 3D-model. The plot above shows view from the point of view of the infrared cameras. The small cylinders are located on the desk to mark the borders of the intended interaction volume

3. PROBLEM STATEMENT

Our goal is to develop and implement an ergonomic and intuitive oblique planar slicer for 3D medical data such as CT and MRI data. The target user group of this application will be the medical doctors and 3D technologists who need to investigate such volumetric data for diagnostic purposes. MDs typically take arbitrary oriented oblique slices of the 3D data to reconstruct the 3D object in their minds. The proposed HCI is aimed at improving this slicing with respect to time and accuracy.

The application developed in this thesis consists of a passive object that we call "3D Stylus" and a stereo camera. Our approach is to compute the orientation of the *3D Stylus* in 3D space by using stereo camera and use its orientation as the normal vector of the slicing plane. The designed system is expected to have following specifications:

1. An intuitive user interface
2. A real time system for stereo processing
3. Continuous and smooth 3D motion reconstruction

Considering all of these criteria, our main objective is building a real time stereo tracking system and 3D oblique volume slicing application. The most challenging requirement among the ones listed above is the 3D motion reconstruction. Moreover after reconstruction, system faces with two sources of error that it has to tackle with: noise to due the imaging system and the natural random motion of human hand.

4. SYSTEM

We will try to cover algorithm and working mechanism of our design in this chapter. Basically, our system can be divided into five main parts namely, image acquisition, 2D processing, 3D processing, temporal filtering and visualization. First part is about taking two images from stereo camera in real time simultaneously. Second part explains all image processing works applied to each image taken from cameras separately, including segmentation,line fitting and so on. Third part will be 3D processing part and *3D Stylus* reconstructed as two 3D points in the space. Fourth part will be temporal filtering part. Since the observation(3D points) obtained end of 3D processing are noisy, they need to be filtered so as to get rid of shaky results during visualization. This part is especially important to obtain convenient interface for users. Otherwise, tremble of the slicing plane will be problem for users. Two different filtering works will be represented in the filtering part. Last part of our system is visualization. Although visualization part is not most critical part of our design, it is sine qua non to compete project. The designed visualization module also will be used for user tests. In the below all these parts are explained in detail. Firstly, we will show overall system's flow chart in the Figure 4.1;

4.1. 2D processing

Each image simultaneously acquired from a stereo camera pair is preprocessed independently to identify the two endpoints of the *3D Stylus*. 2D processing is composed of two stages, namely the segmentation of the *3D Stylus* and the detection of the end points. These two stages are explained in detail in the following:

4.1.1. Segmentation

The *3D Stylus* will be used in a controlled environment, such as the radiology reading room. So, we assumed that the background in the acquired images is a flat black sheet with the *3D Stylus* in the foreground significantly brighter. Based on this

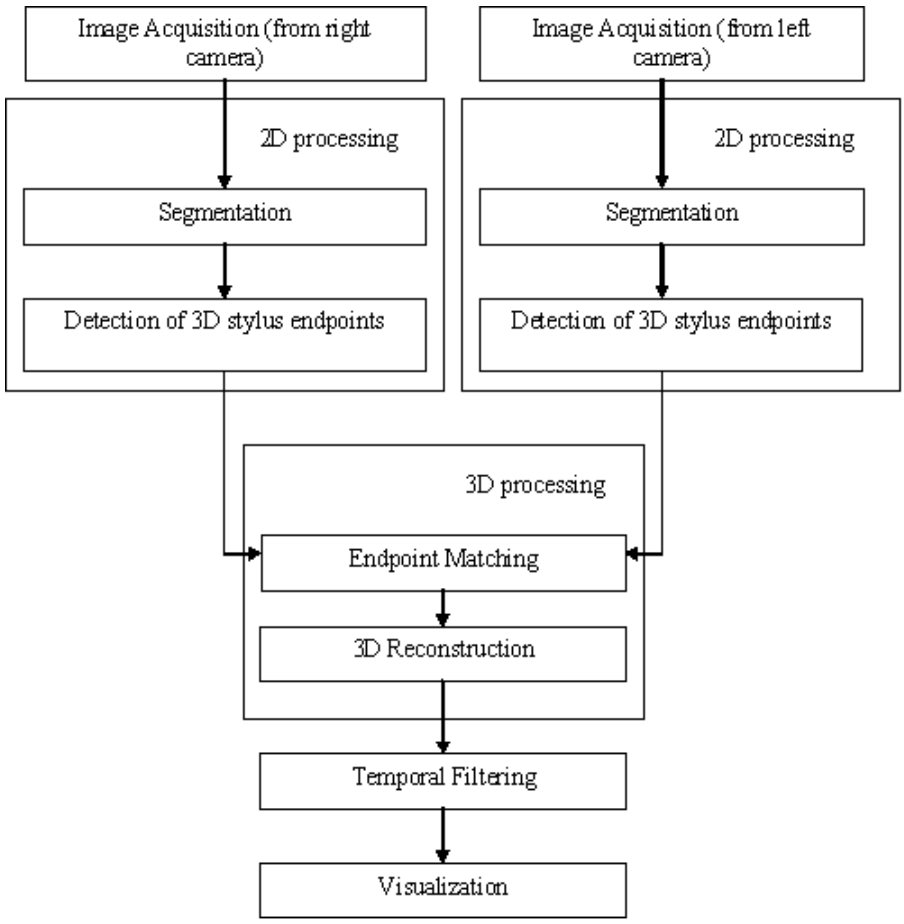

Figure 4.1. System Flow Chart 1

assumption, we initially segmented each image by thresholding with respect to the pixel intensities. Let $I(p_i)$ be the i^{th} pixel of image I located at position $p_i = [x_i, y_i]$,

$$\hat{I}(p_i) = \left\{ \begin{array}{ll} 1 & , I(p_i) > 220 \\ 0 & , \text{otherwise} \end{array} \right\} \tag{4.1}$$

where 220 is an empirically determined intensity threshold. $\hat{I}$ is the binary image computed. In general, there are multiple disconnected regions of 1's in $\hat{I}$. Each region is labeled using the connected component analysis. One of them is the silhouette of the *3D Stylus*. To identify the *3D Stylus*' silhouette, we first filter out the regions with an area smaller than a predetermined threshold, which is empirically set to be 30 pixels.

However, this is not sufficient as there may be regions unrelated to the *3D Stylus* but with a significant area. The final identification of the *3D Stylus* is achieved by Principal Component Analysis (PCA) of the spatial distribution of pixels in each region. Let $p_{ij} \in \Re^{2\times 1}$ be the position of the i^{th} pixel in the j^{th} region, then the covariance matrix of the distribution of all pixels in the j^{th} region is given by,

$$C_j = \frac{1}{K}\sum_{i=1}^{K} p_{ij}p_{ij}^T \tag{4.2}$$

where K is the number of pixels in the j^{th} region. C_j is a symmetric positive semi-definite matrix whose eigenvalues give the variance of the distribution of pixels positions along the corresponding eigenvector. The silhouette of the *3D Stylus* is expected to be a rectangle, which corresponds to a highly anisotropic distribution of the pixels. This is equivalent to a large difference between the two eigenvalues of C_j. Let $\lambda_{j,min}$ be minimum eigenvalue of C_j. Defining $\kappa_j = \frac{\lambda_{j,max}}{\lambda_{j,min}}$, we marked the region with the largest κ as the silhouette of the *3D Stylus*.

Having determined the j^{th} region that corresponds to the *3D Stylus*, a line is fit to all points in that region using the least squares line fitting(LS) [13]. The error term for LS line fitting is defined as

$$E_j = \sum_i (p_{ij}^y - ap_{ij}^x - b)^2 \tag{4.3}$$

where p_{ij}^x and p_{ij}^y are x and y component of p_{ij} respectively, a and b are the constants of the line equation ($y = ax + b$), which needs to be computed.

The steps above are applied to the first frame of both cameras independently. For the following frames, PCA analysis used to identify the *3D Stylus* silhouette may not work because the *3D Stylus* may be perpendicular to the camera plane. So we labeled connected components whose center of mass is closest to the center of mass of the *3D Stylus* silhouette in the previous frame, as the sought silhouette. If it is expressed in analytic form: Let $\underline{m_j}$ be center of mass of the j^{th} component and $\underline{\tilde{m}}$ be the center

of mass of the connected component which represents the *3D Stylus* in the previous frame. The result of $arg(min_j \parallel (\underline{m}_j - \underline{\tilde{m}}) \parallel)$ gives the connected component label which represents the current *3D Stylus* silhouette.

Initially, the *3D Stylus* must be parallel to plane and completely inside the FOV of both cameras. Otherwise, in the initial stage the wrong connected component may be selected as the *3D Stylus* and all tracking algorithm may fail.

4.1.2. Identification of Endpoints

Since tracking of all points of the *3D Stylus* is impossible, we need some salient points of the selected connected component to track. The most convenient points for tracking of the *3D Stylus* are its endpoints. The endpoints are defined as the furthest away point pair on the line segment fit to the segmented silhouette of the *3D Stylus*. Let $\mathbf{EP}_1^i$,$\mathbf{EP}_2^i$.

Let the j^{th} component be the *3D Stylus* silhouette $\mathbf{p}_i \in \Re^{Nx2}$ represents N points which are in the j^{th} connected component and on the line fit to it. We marked the furthest away point pair in $\hat{\mathbf{P}}_j$ as the endpoints, $\mathbf{EP}_1$ and $\mathbf{EP}_2 \in \Re^{2x1}$ be the two end points of detected on the image of the i^{th} camera, $i = 1, 2$

4.2. 3D Processing

4.2.1. Endpoint Matching

The two endpoints pair from two cameras are needed to be matched prior to 3D construction. Because we have only 4 points to match, there are only two possible matching. Orientation information of each *3D Stylus* silhouette is used to determine which matching is correct. Two vectors are defined in each images by choosing either $\mathbf{EP}_1^i$ or $\mathbf{EP}_2^i$ as the origin. Let these vector be $\mathbf{v}_1 = \mathbf{EP}_1^1 - \mathbf{EP}_2^1$ and $\mathbf{v}_2 = \mathbf{EP}_1^2 - \mathbf{EP}_2^2$.

Since the stereo cameras used are positioned close to each other, their viewpoints

are similar. This means that if $\mathbf{EP}_1^1$ $\mathbf{EP}_1^2$ are corresponding points, then $\mathbf{v}_1.\mathbf{v}_2 > 0$, i.e. they are more or less loss similarity oriented. So our matching works as follows:

$$\begin{aligned} \mathbf{v}_1.\mathbf{v}_2 > 0 \quad &, (\mathbf{EP}_1^1, \mathbf{EP}_2^1) \longleftrightarrow (\mathbf{EP}_1^2, \mathbf{EP}_2^2) \\ \mathbf{v}_1.\mathbf{v}_2 < 0 \quad &, (\mathbf{EP}_2^1, \mathbf{EP}_1^1) \longleftrightarrow (\mathbf{EP}_1^2, \mathbf{EP}_2^2) \end{aligned} \tag{4.4}$$

4.2.2. 3D Reconstruction

3D reconstruction implicitly consists of two parts. First one is point pair search part, which we search for a point and its correspondence on the other camera. The second part is camera part which we compute 3D coordinate of point pair with the help of camera projection matrix. The first part is cover in the section 4.1 and now we are ready to compute 3D coordinates of points under unknown correspondence. Firstly basics about 3D reconstruction will be covered and then our computation will be explained. Difference of our computation comes from specifications of our stereo camera. It is very helpful because the consumed time for computations reduce dramatically.

Stereo processing is covered by a geometry called epipolar geometry in literature. The epipolar geometry is the intrinsic projective geometry between two views. Fundamental matrix($\underline{\mathbf{F}}$) or essential matrix can encapsulate this geometry. In the remaining of this part, epipolar geometry will be explained using $\underline{\mathbf{F}}$. Before explaining the fundamental matrix and its functionality, a brief explanation about essential matrix and the fundamental matrix difference can be helpful.

Despite the fact that the fundamental matrix is proposed later, it is general form of essential matrix [21]. Essential matrix assumes camera's intrinsic and extrinsic parameters are known separately, where the fundamental matrix does not. Therefore programmer can save himself from an effort to compute intrinsic and extrinsic parameters separately by using the fundamental matrix. That is why essential matrix is not preferred stereo processing applications. For more information about essential please refer [19].

Point and line in the remaining part will be defined using homogeneous coordinate system. A point in the 2D homogenous coordinates is represented as $\mathbf{p} = [p_x, p_y, 1]^T$. Let $ax + by + c = 0$ be line equation, The line could be represented in homogeneous coordinates as $\mathbf{l} = [a, b, c]^T$. The "primed" variables are used for the right camera image plane related variables, whereas the "unprimed" ones are used for the left camera image plane related variables. Capital letters are used for the point coordinates in 3D.

The fundamental matrix represents a transformation between a line and a point. A point on one image plane correspond to a line on the other image plane. From now on we will call this corresponding line, epipolar line of that point. This relation is depicted in Figure 4.2: the line $\mathbf{l}$ is epipolar line of $\mathbf{p}$. This figure also depicts the basic concept of stereo image processing and variables where we used in this section. Detail of variables on the figure are:

$\mathbf{P}$ is the 3D coordinates of the a point imaged, $\mathbf{p}$ is the $\mathbf{P}$'s image on the left image plane and $\mathbf{p}'$ is $\mathbf{P}$'s image on the right camera. $\mathbf{C}$ and $\mathbf{C}'$ are the centers of left and right camera respectively. $\mathbf{e}$ and $\mathbf{e}'$ also represents epipolar points on each camera plane.

Epipolar geometry states that the image of $\mathbf{P}$ on either one of the two image planes, has to be on the line defined by the intersection of the epipolar plane with that image plane. This line passes through $\mathbf{p}$ and $\mathbf{e}$ points for the left image plane in Figure 4.2[5] and $\mathbf{p}'$ and $\mathbf{e}'$ points for right image plane. Knowing $\mathbf{p}$,$\mathbf{ec}$, we can define the epipolar plane. So we can find the intersection of the epipolar plane with the left image plane $\mathbf{l}'$, $\mathbf{p}'$ is sought on $\mathbf{l}'$.

All constraint related with 3D vision called "epipolar constraints" can be represented by the fundamental matrix. In the below the fundamental matrix will be explained in detail.

[5]http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT10/node3.html

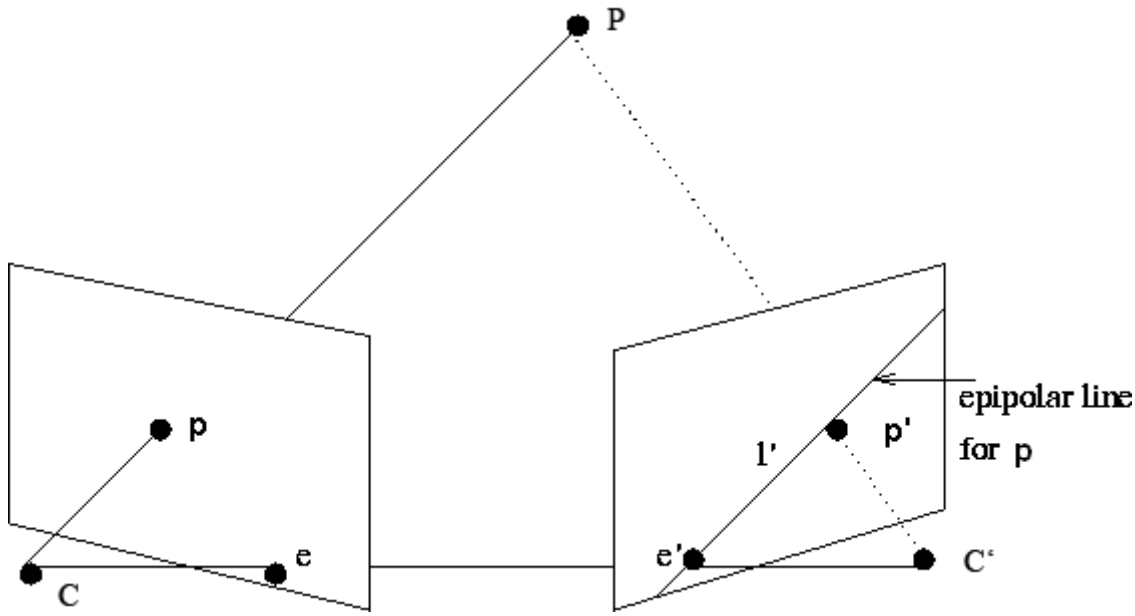

Figure 4.2. Epipolar

4.2.2.1. The Fundamental Matrix. The fundamental matrix's transformation can be written analytically as follows:

$$\underline{l}' = \underline{\mathbf{F}}\mathbf{x} \tag{4.5}$$

In the below we will try to express $\underline{\mathbf{F}}$ in terms of $\mathbf{e}$s and $\underline{\mathbf{T}}$s. Let $\underline{\mathbf{T}}$ be the projection matrix of the left camera and $\underline{\mathbf{T}}'$ that of the right camera. $\underline{\mathbf{T}}\mathbf{P} = \mathbf{p}$ is called projection equation for the left camera.

In the below the fundamental matrix equation will be written in terms of projection matrix and epipolar points to understand its derivation. $\mathbf{l}'$ could be expressed like $\mathbf{l}' = \mathbf{e}' \times \mathbf{p}'$ because both $\mathbf{p}'$ and $\mathbf{e}'$ are on the $\mathbf{l}'$. From Equation 4.5 $\mathbf{l}'$ also equals to

$\underline{\mathbf{F}}\mathbf{p}$. Using these two equations:

$$\underline{l}' = \underline{e}' \times \underline{p}' \tag{4.6}$$

$$\underline{l}' = (\underline{\underline{T}}'\underline{C}) \times (\underline{\underline{T}}'\underline{P}) \tag{4.7}$$

$$\underline{\underline{F}p} = (\underline{\underline{T}}'\underline{C}) \times (\underline{\underline{T}}'\underline{\underline{T}}^{+}\underline{p}) \tag{4.8}$$

$$F = [e]_{\times}T'T^{+} \tag{4.9}$$

where T^{+} is pseudo-inverse of$T$ and $[\mathbf{e}_{\times}]$ denotes the skew symmetric matrix such that $[\mathbf{e}_{\times}]\mathbf{b} = \mathbf{e} \times \mathbf{b}$ is the cross-product of the vectors e and b. Consequently, $\underline{\mathbf{T}}$, $\underline{\mathbf{T}}'$ and $\mathbf{e}'$ are sufficient to find the fundamental matrix. Although the fundamental matrix could be computed using these variables, for most of the cases $\mathbf{e}'$, $\underline{\mathbf{T}}$ and $\underline{\mathbf{T}}'$ are unknown. In general these unknowns are computed using an estimated $\underline{\mathbf{F}}$. In most of the stereo applications the fundamental matrix is computed using the basic epipolar constraints. One of the basic constraints which helps to find the fundamental matrix is given as follows:

For a point $\mathbf{p}$ on a line $\mathbf{l}$, $\mathbf{p} \cdot \mathbf{l} = 0$. So,

$$\mathbf{p}' \cdot \mathbf{l}' = 0 \tag{4.10}$$

$$\mathbf{p}' \cdot (\underline{\mathbf{F}}\mathbf{p}) = 0 \tag{4.11}$$

$$(\mathbf{p}')^{T}\underline{\mathbf{F}}\mathbf{p} = 0 \tag{4.12}$$

$\underline{\mathbf{F}}$ is estimated using equation 4.12 as explained in the following.

4.2.2.2. 3D reconstruction. In Section 4.2.2.1, the fundamental matrix and its importance is mentioned. As expressed at the end of the section most of the time the fundamental matrix is used as a tool which assists to compute the camera matrix. The explanation given below is about calculation of a 3D point under the condition that two corresponding points are known on a stereo camera. Then bumblebee stereo camera equations will be revealed. Because bumblebee stereo camera is a product of

Point Grey Co., It is calibrated and equations are derived according to this calibration by the company before.

The standard 3D reconstruction can be done by following 3 basic steps:

1. Computation of the Fundamental Matrix
2. Computation of the Camera Matrix
3. Triangulation

In the first step; the fundamental matrix $\underline{\mathbf{F}}$ is calculated from given eight point correspondences. $\underline{\mathbf{F}}$ is a 3×3 matrix and it has 9 coefficients. Eight point is the minimum number of correspondence pairs to compute $\underline{\mathbf{F}}$'s entries up to a scale because we assume $F_{3\times3} = 0$. This calculation is called calibration in literature. If the orientation of cameras is fixed with respect to each other, there is no need to further calibrate cameras.

In the second step, a pair of camera matrices $\underline{\mathbf{T}}$ and $\underline{\mathbf{T}}'$ corresponding to the fundamental matrix $\underline{\mathbf{F}}$ are computed. Selecting the world coordinates origin as the first camera coordinates origin, we get

$$\mathbf{C} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.13}$$

Assuming unit focal length, the camera matrix of the left camera has to be $\underline{\mathbf{T}} = [I_3|0]$ where I_3 is 3×3 identity matrix. It is proven in [16] that the following equation can be used to find the right camera matrix:

$$\underline{\mathbf{T}}' = [[\mathbf{e}_\times]\underline{\mathbf{F}}|\underline{\mathbf{e}}'] \tag{4.14}$$

It is also known that $\mathbf{e}'^T\underline{\mathbf{F}} = 0$ which means all epipolar lines correspond to other camera image points pass through epipolar point. So, $\mathbf{e}'$ could be computed using the fundamental matrix. Therefore we can get both the camera matrices with the help of the fundamental matrix.

Last step of 3D reconstruction is to locate $\mathbf{P}$ in 3D coordinate system using $\mathbf{p}$,$\mathbf{p}'$,$\underline{\mathbf{T}}$ and $\underline{\mathbf{T}}'$. In literature there are several methods to find 3D location of $\mathbf{P}$, such as the following one [19].

For $\underline{\mathbf{T}}$ being the camera projection matrix for left camera,

$$\mathbf{p} = (\underline{\mathbf{T}}\mathbf{P}) \Longleftrightarrow \mathbf{p} \times (\underline{\mathbf{T}}\mathbf{P}) = 0 \tag{4.15}$$

So,

$$\underline{\mathbf{T}}\mathbf{P} = \begin{bmatrix} \underline{\mathbf{t}}_1^T \\ \underline{\mathbf{t}}_2^T \\ \underline{\mathbf{t}}_3^T \end{bmatrix} \mathbf{P} \tag{4.16}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{t}}_1^T\mathbf{P} \\ \underline{\mathbf{t}}_2^T\mathbf{P} \\ \underline{\mathbf{t}}_3^T\mathbf{P} \end{bmatrix} \tag{4.17}$$

$$=> \begin{bmatrix} \underline{\mathbf{t}}_1^T\mathbf{P} \\ \underline{\mathbf{t}}_2^T\mathbf{P} \\ \underline{\mathbf{t}}_3^T\mathbf{P} \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0 \tag{4.18}$$

For which we get,

$$y(\underline{\mathbf{T}}_3^T\mathbf{P}) - (\underline{\mathbf{T}}_1^T\mathbf{P}) = 0 \tag{4.19}$$

$$x(\underline{\mathbf{T}}_3^T\mathbf{P}) - (\underline{\mathbf{T}}_2^T\mathbf{P}) = 0 \tag{4.20}$$

Equation 4.19 and 4.20 are derived for left camera. repeating the same steps for the

right camera and combining all four equations into a matrix equation, we get,

$$\underline{\mathbf{A}}\mathbf{P} = \begin{bmatrix} x\underline{\mathbf{T}}_3^T - \underline{\mathbf{T}}_1^T \\ y\underline{\mathbf{T}}_3^T - \underline{\mathbf{T}}_2^T \\ x'\underline{\mathbf{T}}_3'^T - \underline{\mathbf{T}}_1^T \\ y'\underline{\mathbf{T}}_3'^T - \underline{\mathbf{T}}_2^T \end{bmatrix} \mathbf{P} = 0 \tag{4.21}$$

Let $\underline{\mathbf{A}} = \mathbf{U}\underline{\mathbf{D}}\mathbf{V}^T$, then the solution minimizing error must be parallel to the right singular vector corresponding to the minimum singular value.

Bumblebee Stereo camera system consists of two two cameras in a metal box. Because they are fixed inside the box , no calibration routine is needed. Another important property of the bumblebee stereo camera is that the transformation between image planes can be represented by only translation. These two properties make the 3D reconstruction equations simple. These reconstruction equations are given by the producer of the bumblebee camera.

Z axis is the perpendicular axis to image plane and can be computed using:

$$Z = fB/d \tag{4.22}$$

where Z is the distance along the camera Z axis f is the focal length (in pixels) B is the baseline (in meters) d = disparity (in pixels). Disparity is a common term in stereo processing. It means the pixel difference between each image plane for corresponding point pair. Baseline is the distance between center of cameras.

After Z is determined, X and Y can be calculated using the usual projective camera equations:

$$X = uZ/f \tag{4.23}$$

$$Y = vZ/f \tag{4.24}$$

where $[X, Y, Z]^T$ is the real 3D position, $[u, v]$ is the pixel location in the 2D image measured;

$$u = \text{center column index - column index} \tag{4.25}$$

$$v = \text{center row index} - \text{row index} \tag{4.26}$$

To sum up, 3D coordinate of a point can be computed using bumblebee, as long as we know the disparity.

4.2.3. Construction of 3D Stylus Orientation

Calculation of 3D coordinates of the endpoints of the *3D Stylus* is followed by construction of a 3D vector which represents *3D Stylus*. 3D vector is defined to be the vector connecting two 3D endpoints. The only problem in this part is ambiguity of vector's sign. Choosing its sign arbitrarily on the first frame, the sign of the 3D vector on consecutive frames is determined so as to minimize the angle between consecutive 3D vectors.

Let $\mathbf{V}_0$ be the vector calculated using the current frame and $\tilde{\mathbf{V}}$ is the vector calculated in the previous frame.

$$\begin{aligned} \mathbf{V}_0 \cdot \tilde{\mathbf{V}} > 0 \quad &\text{, Orientation of 3D vector is correct} \\ \mathbf{V}_0 \cdot \tilde{\mathbf{V}} < 0 \quad &\text{, Revert the 3D vector} \end{aligned} \tag{4.27}$$

4.3. Temporal Filtering

Temporal filtering is required due to the non-stationary hand motion and other noise sources. As a result, using the reconstructed 3D vector directly as the normal of our slicer result a shaky plane. Orientation information needs to be filtered. Two different methods are used to filter out the noise, namely the Kalman Filter and the Particle Filter. In the below we will explain details of these methods and how we use

them.

4.3.1. Kalman Filter

The Kalman Filter [14] is a recursive filtering technique which is proposed by Rudolph Kalman in 1960. Kalman filter estimates the state of a dynamic system from a series of incomplete and noisy measurements [17]. State could be a scalar or a vector of real numbers. Kalman filter defines the system dynamic and the observation relation according to the expressions below,

$$\mathbf{x}_k = \underline{\mathbf{F}}_k \mathbf{x}_{k-1} + \underline{\mathbf{B}}_k U_k + \mathbf{w}_k \tag{4.28}$$

$$\mathbf{Z}_k = \underline{\mathbf{H}}_k \mathbf{x}_k + \mathbf{v}_k \tag{4.29}$$

Where;

$\underline{\mathbf{F}}_k$: State Transition matrix
$\underline{\mathbf{B}}_k$: Control input matrix
$\underline{\mathbf{H}}_k$: Observation matrix
$\mathbf{w}_k$: Process Noise, N(0, Q_k)
$\mathbf{v}_k$: Observation Noise, N(0, R_k)
$\mathbf{x}_k$: State Vector

According to Equation 4.28, the system state is linearly dependent to the previous state. On the other hand, the dynamic equation, Equation 4.29, states that there is a linear relationship between the state and the observation. In some cases this linear approach will not be applicable. There are some approaches proposed in literature to solve non-linear systems in a way similar to the classical Kalman filtering. There are two common approaches for non-linear systems, the Extended Kalman filter (EKF) and the Unscented Kalman Filter (UKF) [18]. EKF gives particularly poor performance on highly non-linear functions because only mean is propagated through the non-linearity. Because UKF uses a deterministic sampling technique, and UKF captures the true mean and covariance more accurately, as well as removing the re-

quirement to analytically calculate Jacobian because it uses a deterministic sampling technique.We used classical kalman filter in this thesis.

Kalman filter consists of two phases. The first one is the prediction phase where we predict the next value of state vector with the help of the system dynamics and the previous state value. The second is the correction phase where we correct our prediction using observations. Both stages can be expressed analytically as follows:

Prediction Stage:

$$\hat{\mathbf{x}}_k = \underline{\mathbf{F}}_k \mathbf{x}_{k-1} + \underline{\mathbf{B}}_k \mathbf{U}_k (\text{State prediction}) \tag{4.30}$$

$$\hat{\mathbf{P}}_k = \underline{\mathbf{F}}_k \underline{\mathbf{P}}_{k-1} \underline{\mathbf{F}}_k^T + \mathbf{Q}_k (\text{Covariance prediction}) \tag{4.31}$$

Correction Stage:

$$y_k = Z_k - H_k \hat{x}_k (\text{Innovation}) \tag{4.32}$$

$$S_k = H_k \hat{P}_k H_k^T + R_k (\text{Innovation covariance}) \tag{4.33}$$

$$K_k = \hat{P}_k H_k^T S_k^{-1} (\text{Kalman Gain}) \tag{4.34}$$

$$x_k = \hat{x}_k + K_k y_k (\text{Corrected Estimate}) \tag{4.35}$$

$$P_k = (I - K_k H_k) \hat{P}_k (\text{Corrected Estimate}) \tag{4.36}$$

Kalman filter assumes that the distribution of state vector is a multivariate gaussian and that the noise is generated by independent zero-mean gaussian random process.

Kalman filter is an iterative technique and it is a quite fast algorithm. Consequently, it is widely used for real time signal processing applications in computer vision. In our application, we have applied Kalman filter to the observation of the 3D vector ($\mathbf{Z}$).

The state vector is

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.37}$$

where each entry corresponds to a dimension of the 3D vector representing the *3D Stylus*. The system dynamics is modeled as a random walk:

$$\mathbf{x}_k = \underline{\mathbf{F}_k}\mathbf{x}_{k-1} + \mathbf{w}_k \tag{4.38}$$

where $\underline{\mathbf{F}_k}$ is a 3×3 identity matrix and $\mathbf{w}_k$ is a gaussian noise, $N(0,0.0001)$. The observation is also $\mathbf{x}$. Therefore, the relation between $\mathbf{x}$ and the observation, $\mathbf{z}$ differs only by a noise $\mathbf{v}_k$, $N(0,0.001)$:

$$\mathbf{z}_k = \underline{H_\mathbf{k}}\mathbf{x}_k + \mathbf{v}_k \tag{4.39}$$

where $\underline{\mathbf{H}_k}$ is a 3×3 identity vector and $\mathbf{Z}_k$ is our observation vector.

4.3.2. Particle Filter

We have used the kalman to filter our observations,i.e. the 3D components of the unit normal. Nevertheless in kalman filtering we have been using a redundant dimension. The unit normal is representable with a two dimensional vector. The spherical coordinates (θ and ϕ) angles are enough to represent the unit vector. Kalman filter is based on gaussianity assumption. yet, the normal vector parameter (θ, ϕ) are not necessarily gaussian. Especially under the assumption of smooth motion, θ and ϕ sequences deviate from gaussianity. Furthermore, θ and ϕ's domain is the unit sphere. Consequently, we need a pdf defined on the unit sphere that allow us to model the expected θ and ϕ distributions based on past observations. Kent distribution is an appropriate choice as it is parametrically defined on the unit sphere [15]

4.3.2.1. Kent Distribution. It could be defined simply as the analogue of the bivariate normal distribution on the two-dimensional unit sphere. Analytically,

$$F(x) = \frac{1}{c(\kappa,\beta)} exp(\kappa\gamma_1 x + \beta[(\gamma_2 x)^2 - (\gamma_3 x)^2]) \tag{4.40}$$

where

$$c(\kappa,\beta) = \int_0^{\pi}\int_0^{2\pi} exp(\kappa\cos\theta + \beta\sin^2\theta\cos 2\phi)\sin\theta d\phi d\theta \tag{4.41}$$

κ the concentration of the distribution

β the ellipticity of the contours of equal probability

γ_1 the mean direction

γ_2 the major axis of ellipse

γ_3 the minor axis of ellipse

In multivariate gaussian distribution we use covariance matrix contains information both about concentration of distribution and isotropy. Differently, kent distribution needs 4 variable to explain these properties. κ is functioned like variance in 1D gaussian distribution and only determines how peek the distribution is. γ_2 and γ_3 determines main direction of isotropy. β shows how isotropic the data is along the main directions.

4.3.2.2. Particle Filter. The variable we want to know throughout the process is described by an n-dimensional state vector x. assuming that we are not able to know the exact state which is tracked using a probability function $p(x)$. Observations will change over time and the function evolves to represent new state. The dynamics of

state vector is:

$$x_t = f(x_{t-1}, w_{t-1}) \tag{4.42}$$

where w_{t-1} is noise of system with a known distribution. The probability function which correspond to difference equation above is a relation between $p(x_t)$ and $p(x_{t-1})$:

$$p(x_t) = \int p(x_t|x_{t-1})dx_{t-1} \tag{4.43}$$

where $p(x_t|x_{t-1})$ is called process density. Since Eq. 4.43 is only dependent the previous state, it is a markov process also.

In particle filter we also assume that observations are related to state vector. Let z_t be observation vector at time t.

$$z_t = h(x_t, v_t) \tag{4.44}$$

where v_t is observation noise and h is a function which maps state vector to observation. State estimation needs to be estimated every time step using observation. as mention at the beginning, we are after the probability function of state x, because there is an uncertainty in state. Let Z_t be history of observations, $\{z_1, ...z_t\}$. Z will be added to the probability function: $p(x_t|Z_t)$ and $p(x_t|Z_{t-1})$. The former is called posterior and the latter is called prior distribution. Because we want to know posterior distribution, it is rewritten using Bayes rule:

$$\begin{aligned} p(x_t|Z_t) &= \frac{p(z_t|x_t, Z_{t-1})p(x_t|Z_{t-1})}{p(z_t|Z_{t-1})} && (4.45)\\ &= kp(z_t|x_t, Z_{t-1})p(x_t|Z_{t-1}) && (4.46)\\ &= kp(z_t|x_t)p(x_t|Z_{t-1}) && (4.47) \end{aligned}$$

where k is a normalization coefficient. The proof of simplification from 4.46 to 4.47 is given in [20].

In general, the derived equation below is a recursive bayesian filtering and it is too complex to evaluate in a closed form. So, stochastic sampling methods are used to obtain current density function. Factored sampling is used to find an approximation of function:

$$g(x) = g_2(x)g_1(x) \tag{4.48}$$

a set of sample $s = \{s^1 i ... s^N\}$ are drawn from $g_1(x)$ randomly. Then a weight is calculated for every sample which we call them particle according to the rule below using $g_2(x)$:

$$\pi^i = \frac{g_2(s^i)}{\sum_1^N g_2(s^i)} \tag{4.49}$$

where π^i is the weight of the i^{th} particle. As $N \longrightarrow \infty$, the distribution gets closer to $g(x)$. After obtaining new distribution, particles are resampled according to the rule below.

Let $cf(x) = \sum_{i=1}^{x} \pi^i$ be cumulative function current particles. New s array will be

$$s^i = argmax_j(cf(j) < i/N) \text{ for all i=1...N} \tag{4.50}$$

In our case, a distribution on a unit sphere is sought because any point on it capable of representing 3D vector. Therefore our observation is a point on sphere as well as being a vector. As mentioned in section 4.3.2.1, Kent distribution is a suitable distribution on the unit sphere with 5 parameters. 3 of the parameters are calculated with an heuristic approach. These parameters are γ_1, γ_2 and γ_3. γ_1 is assumed to be the same our observation. γ_2 is taken as our velocity vector which we computed as difference between current and previous observation. γ_3 is the cross product of γ_1 and γ_2. Remaining two parameters (α,β) computed using particle filter.

Initially, we draw (α,β) particles in the space uniformly. Then, initial particles are spread using system dynamic. System dynamic is assume as a random walk model. The noise in the random walk model we used is: $N(0,0.001)$.

In the second step, observation is used to compute weights of particles. The function for weights is Kent distribution. Three parameters (γ_1,γ_2,γ_3) are comes from our heuristic approach and two parameters (α,β) are our state vector.

In the remaining steps, the algorithm explained above is strictly applied.

4.4. TWO DIMENSIONAL CONTOUR DRAWING

Contour Drawing is done by projecting 3D points acquired onto a plane defined by user. This is second mode of our system. The other mode was oblique slicing and explained above in detail. The steps shown in the flow chart (Figure 4.1) are followed until the construction of the *3D Stylus* orientation. From that point the steps explained below are followed:

3D points coordinates of two endpoints are taken firstly. Then, one of the points which stays further from camera than the other one is marked as tracking point, because we only need one point to track. The identification of tracking point is followed by definition of plane. The definition is done by three points show by user interactively. These points are upper left corner, origin and lower right corner. The plane is defined using these 3 points as well as origin. Once plane is defined, measured point during the tracking are projected onto the plane.

The Figure 4.3 shows the contour which is used in tests. The Figure 4.4 illustrates the concept. Users are able to draw anything they want on the screen. The black contour do not really exist but put in the picture to make working mechanism clear.

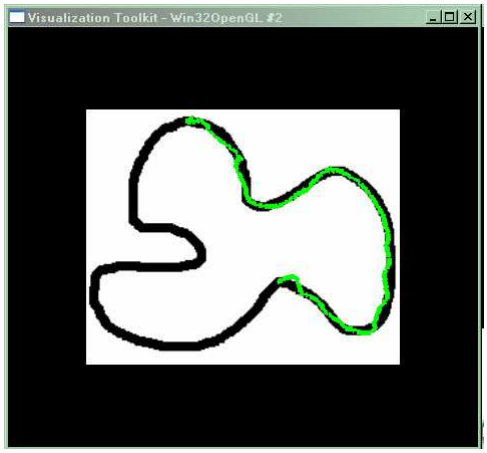

Figure 4.3. A snapshot from drawing screen. The black closed contour is exist initially. The green point are drawn by user using 3D Stylus.

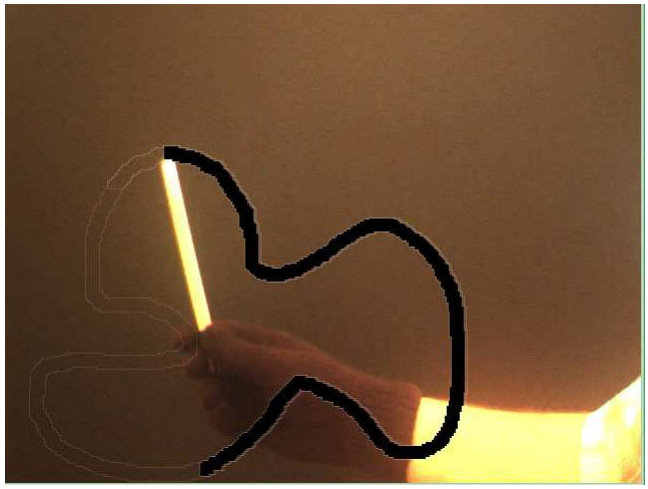

Figure 4.4. A snapshot from stereo camera when 3D stylus is being used. The black contour is drawn there for illustration purpose, it does not really exist.

5. IMPLEMENTATION

A setup is established and software code is written according to steps explained in chapter 4. the Setup is shown in Figure 5.1. The black plane seen on right of the picture has two functionality, black background and hard surface for 2D drawing. The metallic yellow colored box in the middle of the picture is our stereo camera on a tripod. The screen on the left is belongs to the computer which we do our computations and also user can observe visualization for tests.

During the implementation of algorithms, Visual C++ 2005 is used. Tests are done in debug mode. Additionally, software also tested in VC++ 2003 successfully. The C libraries used for programming are given below:

- digiclops : it captures frames from bumblebee camera
- triclops: it converts the captured data from bumblebee format to OpenCV data format
- OpenCV [26]: it realizes basic image processing works such as thresholding, also for all matrix calculation OpenCV matrix structure is used.
- VTK [27]: Visualization of the system is realized with this library.

Default console application settings for VC++ 2005 is taken. Versions of VTK is 5.03 and it is implemented by calling .dlls and no cmake is used in the precompiled stage. Version of digiclops is 2.4 and that of triclops is 3.1. They are obtained from Point Grey Research [22]. openCV is downloaded from [28] and its version is 1.0.

The computer we have used for tests has the following specifications:

- CPU: 2.4 GHZ
- RAM: 2048 Mb
- Operating System: Windows XP

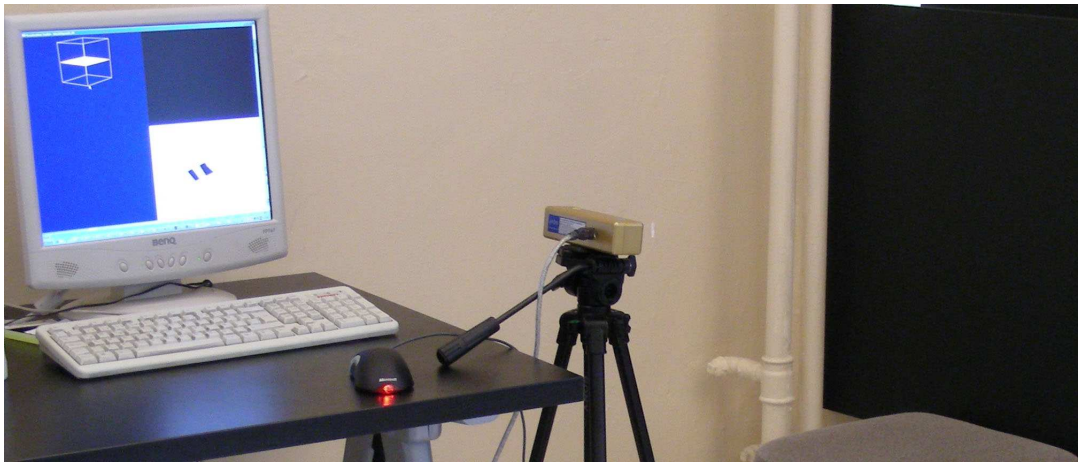

Figure 5.1. Picture of experiment setup. It shows the background, stereo camera and the user screen

The main property of bumblebee camera is its especial structure for stereo processing applications. It has two cameras inside the camera frame and fixed. Since calibrations are done in the stage of production, the equations revealed for 3D coordinate calculation of a point always can be used. Some especial properties of the bumblebee camera are:

- 640X480 resolution
- two vertically aligned color camera in a package
- max. 25 frame/second
- 6 pin firewire interface for computer communication.
- lens with 3.8 mm focal length

6. EXPERIMENT

6.1. Comparison of Kalman and Particle Filter

Kalman filter and particle filter are tested for 3 different cases, perpendicular to camera motion, parallel to camera motion, and non-motion. Tests are realized for the frame length of 200-300. The observations are the θ or ϕ angles of unit vector in the spherical coordinates. A B-spline is fitted to observation data with five control points. This B-spline is accepted as gold standard. The control points of B-spline are beginning,ending, middle point and two more points in the middle of the way between midpoint and the other two end control points.

The error of θ or ϕ angle is computed according to gold standard. Ten point moving average of the error and its variance is computed. The results are represented in Figure 6.1, 6.2 and 6.3 as θ or ϕ angle, the filtered angles, the error and error variance which is measured according to explained procedure above.

Our bayesian based particle filter's mean error is lower than kalman filter in both perpendicular and parallel motion sequences as seen in the Figure. However kalman filter's error variance is lower than particle filter's. Particle filter's noise reduction can be observed in the significant fluctuation part of the motions. On the other hand, Kalman filter is always treat in smooth manner. As can be seen in the Figure 6.1, 6.2 and 6.3 this smoothness costs lag in the time.

The non-motion time sequence has different characteristic. In terms of error variance it is similar to motion part. However particle filter quite successful against periodical fluctuations of observations as seen in the figure.

Since the smoothness of motion (low error variance) is more critical than lag and mean error, provided that the mean error is low enough not to impose any limitation on the practical use of *3D Stylus*, we chose to use kalman filter and conducted the

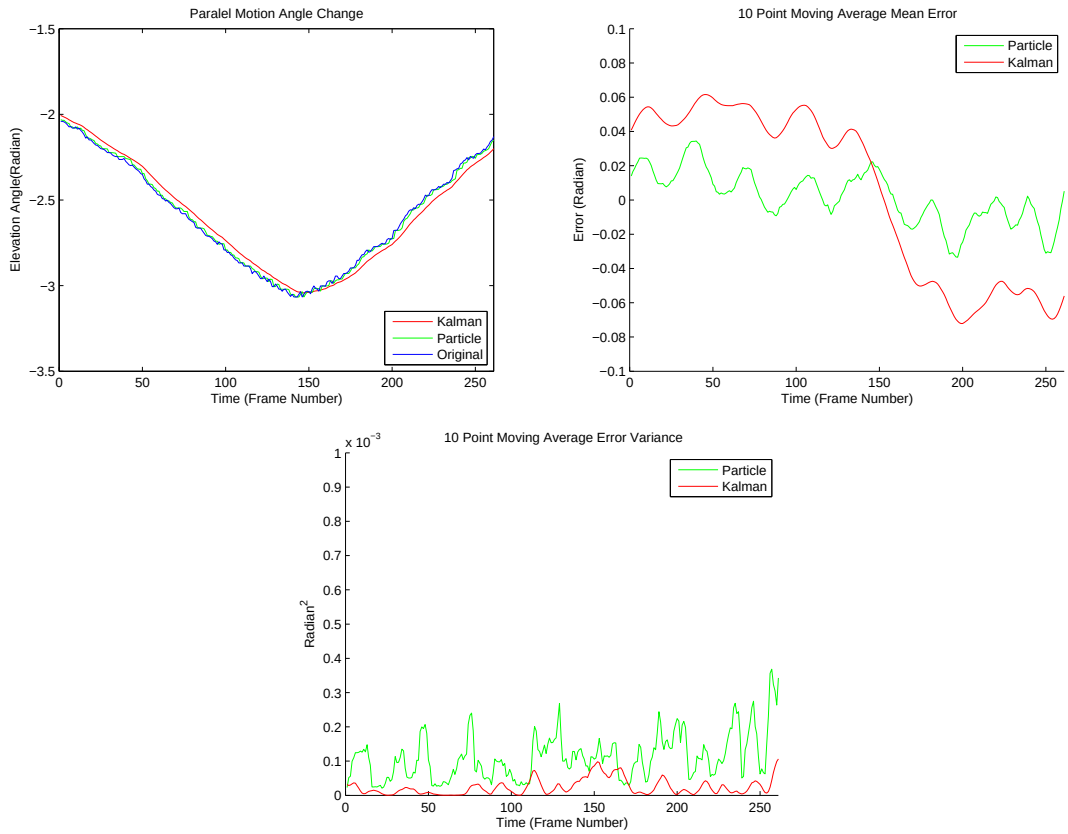

Figure 6.1. Results of Filtering for the motion parallel to the camera plane. The left image on the top illustrates the observed angle and the data after filtering, the right image shows mean error after ten point moving average. The plot at the bottom shows the variance of the error

usability tests with it.

6.2. Usability Test Setup

Performance tests consists of two tests. The first part is about oblique slicing and measures the *3D Stylus*' success in 3D operations. The second part is 2D drawing and measures its success in 2D operations. Each of them compare usability of the *3D Stylus* with that of classical mouse.

In the first test, ten volumetric letters are put into a transparent cube one by one. A plane which represent oblique slicer is also put into the cube. The plane is visible but letter. Figure 6.4 illustrates the screen when test is being done. User is

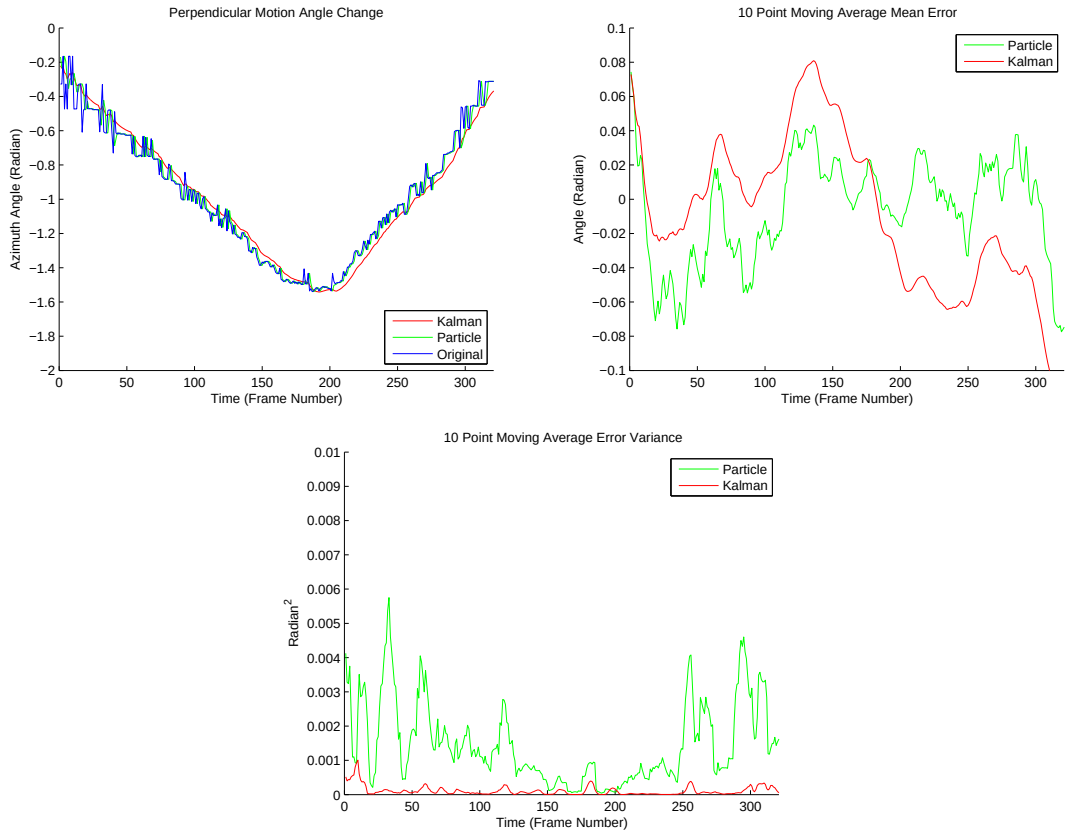

Figure 6.2. Results of Filtering for the motion perpendicular to the camera plane. The left image on the top illustrates the observed angle and the data after filtering, the right image shows mean error after ten point moving average. The plot at the bottom shows the variance of the error

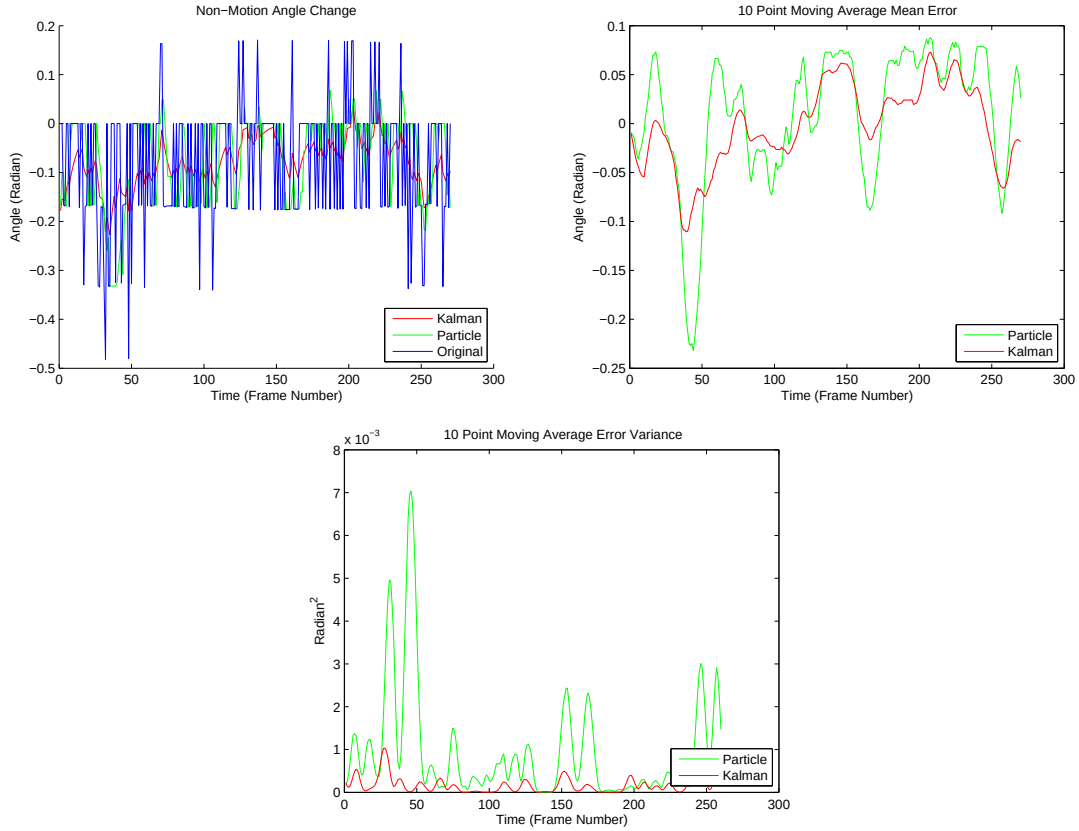

Figure 6.3. Results of Filtering for the non- motion case. The left image on the top illustrates the observed angle and the data after filtering, the right image shows mean error after ten point moving average. The plot at the bottom shows the variance of the error

asked to use the *3D Stylus* and predict the letter by changing the orientation of the slicer. Some cross-sections seems like cross section of a letter other than real letter. In this case user is asked to go on search for real letter. When the real letter is seen on the cross-section plane, user is given next letter. Time is recorded throughout the ten letter sequences.

In the second part of the first test, user followed the same procedure as he did in the first test. However, this time user interface is changed to mouse. Again, the time of prediction of each letter is recorded. We completely took the vtkImagePlaneWidget as mouse manipulation test software so as to compare with our *3D Stylus* software. The only modification on original vtkImagePlaneWidget code is removing of translation.

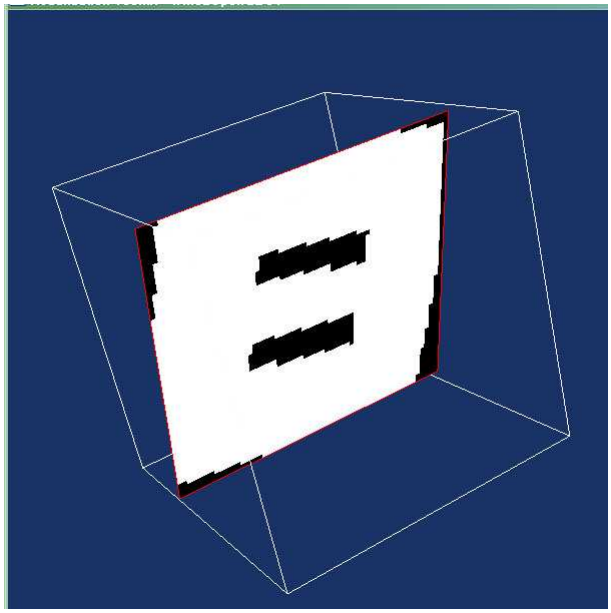

Figure 6.4. Orientation test screen. The white frame shows the limits of transparent cube. The white plane in the middle of the cube is slicing plane and it shows the cross-section of invisible volumetric letter

Therefore user can only rotate slicing plane and an environment is close to our software for *3D Stylus* interface.

The second test tries to evaluate success of user to follow a curve on the screen. This test is very similar to slicing test and consists of two parts. The picture of the curve is given in the Figure 4.3. The drawing task is released on a vertical black plane which can also be seen in Figure 5.1. The curve is drawn by user and total elapsed time is recorded for both the *3D Stylus* and mouse.

The second measurement is the ratio between the number of pixels inside the contour and total number of pixels. We also called it as ratio in the result tables, Table 6.3

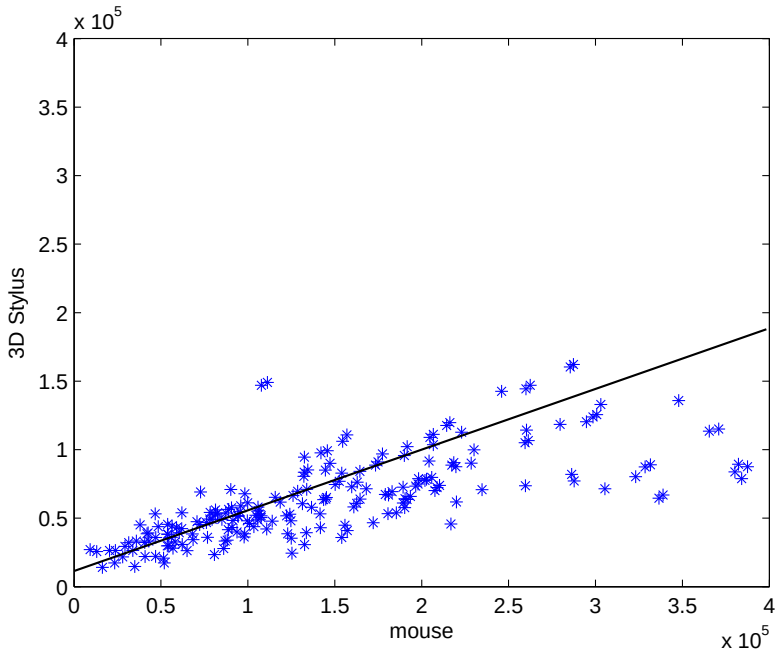

Figure 6.5. This plot is stylus time versus mouse time graph. it illustrates the advantage of stylus in terms of prediction time

6.3. Usability Test Results

In the first test we have tried to understand how successful our *3D Stylus* in slicing applications compare to a mouse. The best parameter illustrating its success is average time of prediction per letter. Twenty people have tested both *3D Stylus* and mouse. The test results for each user is given in the Table 6.3. Except the User7 all users average letter prediction time with *3D Stylus* is better than with mouse. The average prediction time for one letter over 200 samples (20 user 10 letters) is 24946 millisecond for mouse and 10513 millisecond for the *3D Stylus*. There is a big gap between prediction times. The *3D Stylus* is much more successful than the mouse in this experiment. Another thing which illustrates success of *3D Stylus* against the mouse in orientation tests is time plot given in the Figure 6.5. The plot in the figure shows the same letter prediction time with mouse and *3D Stylus*. Therefore one axis is mouse prediction time and the other one is prediction time with *3D Stylus*. If we fit a line using least square sense to the plot, the line equation will be y=0.1x+7922. Therefore the angle of the line will be about 18 degrees. This is also another objective

		MOUSE			STYLUS	
User	Average	Max	Min	Average	Max	Min
User1	25997	88078	3141	14439	42703	1813
User2	22847	54468	1703	9008	29890	1531
User3	33158	83468	2625	8897	35875	1578
User4	38745	140890	1859	8750	26437	2657
User5	21967	41000	1547	8750	26437	2657
User6	21031	52860	1469	7386	16046	1438
User7	11125	31406	1328	14909	79156	2109
User8	37094	94187	5422	11506	51890	1594
User9	38230	125375	2140	8931	24421	2328
User10	21625	57468	1829	11977	39296	2109
User11	26134	67875	1515	10656	45046	1484
User12	21852	65015	1125	9052	28500	1437
User13	9055	34921	1218	5753	16672	1110
User14	20675	85062	1969	11116	49187	1407
User15	20573	67140	1922	7977	21765	2234
User16	17488	77437	1406	9103	29406	2344
User17	30069	68062	2156	12556	47625	1031
User18	28736	69594	1844	16222	43062	1812
User19	17741	51875	141	9694	30141	1640
User20	34789	72750	1656	13586	40078	2891

Table 6.1. The time results of tests for mouse and *3D Stylus*. three different timing is given, maximum,minimum and average prediction time

criteria proves the success of *3D Stylus*.

Another evaluation method we use to understand success of the *3D Stylus* is questionnaire. We have asked the users the following questions and recorded the score they gave:

Question one: How difficult is using *3D Stylus*/Mouse

- 1->very difficult
- 2->difficult
- 3->nor difficult/neither easy
- 4->easy
- 5->very easy

Question two: How successfully did you realized you wishes

- 1->I couldn't do anything
- 2->I couldn't realize most of them
- 3->I realized about half of them
- 4->I realized most of them
- 5->I realized all

Question three: How difficult was the first settings

- 1->very difficult
- 2->difficult
- 3->neither easy nor difficult
- 4->easy
- 5->very easy

Question four: did the system get stuck while you are using

- 1->always
- 2->many times
- 3->sometimes
- 4->rarely
- 5->never

The first question aims at understanding how the user feel when using interfaces. The second question searches for usability. Although users might not feel relax when using, they may do what they want or vice versa. The third question exists to understand whether user feel difficulty when starting. Especially *3D Stylus* wants the user stay stable for a short time at the beginning. The fourth question also point to another problem: getting out side the camera view and stopping of program or limitation of mouse in orientation test and not responding some mouse motions as user expected.

Users replied four questions firstly considering *3D Stylus* then mouse. The numbers in the cells shows the number of people choose the score on the column for the question on the row. Figure 6.6 illustrates the score as pure graph. Each row represents *3D Stylus*

In the second test we asked users to draw the curve by passing on it. The curve is given in the Figure 4.3. The Table 6.3 illustrates the results we recorded. The three parameters we evaluate are ratio, time and length. Ratio means the ratio between the pixels drawn inside the curve and all pixels drawn by user. Time is elapsed milliseconds during the drawing and length is the curve length drawn.

The mean drawing complete time with mouse and *3D Stylus* are 102 and 42 seconds and the mean ratios are 0.74 and 0.80 respectively. Therefore in 2D drawing mouse is more successful in terms of both time and accuracy.

Similar to test one in the second test we again asked the same questions to the users and recorded the score they gave. Figure 6.7 shows the questionnaire results of mouse and stylus experiment. These graphs also report that mouse is more appropriate

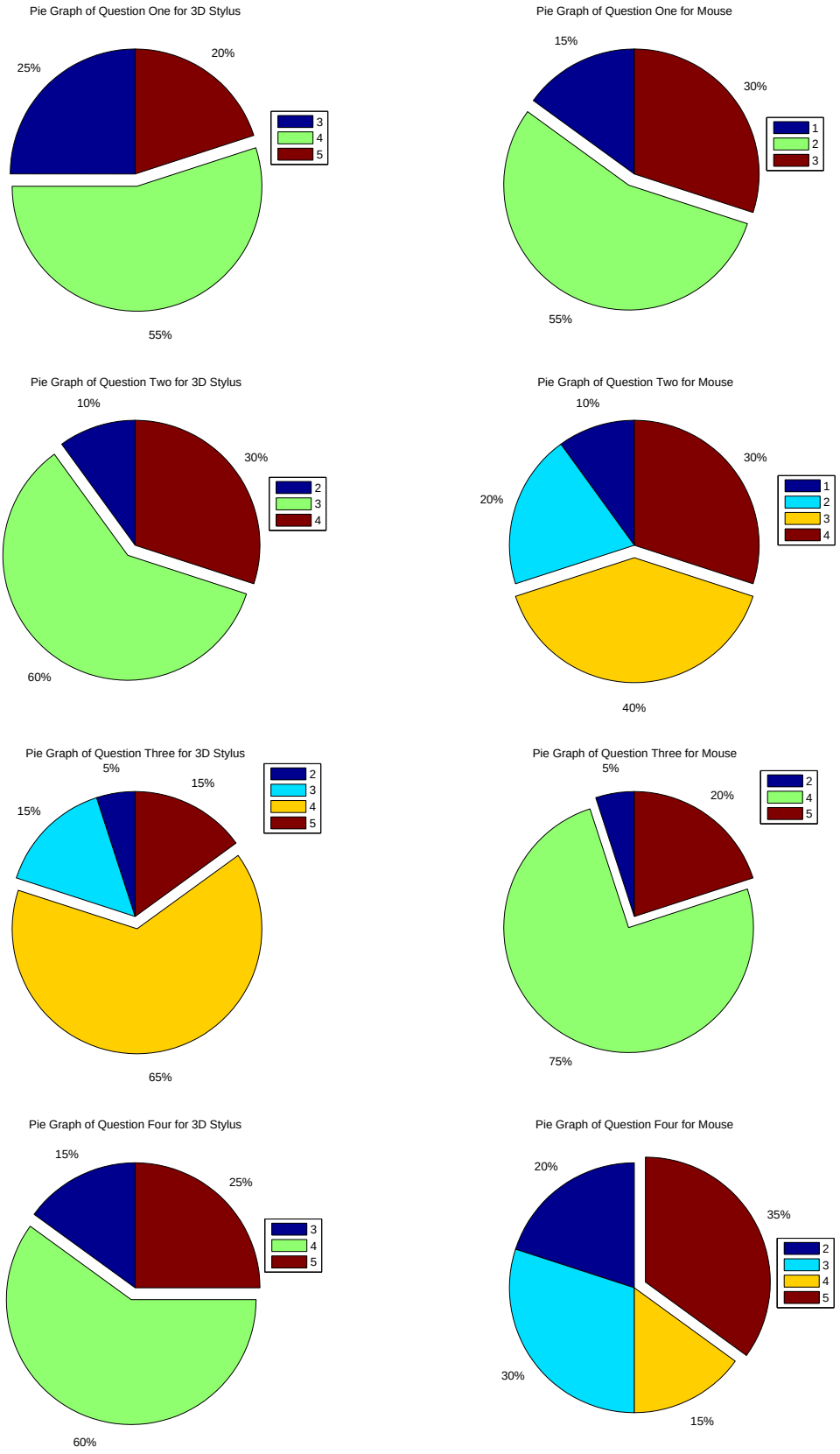

Figure 6.6. Orientation test results. First Column is 3D Stylus Results

User	Ratio	time (ms)	length (pixel)
User1	0,66	87687	1209
User2	0,79	104578	1333
User3	0,91	66250	1038
User4	0,72	123703	1594
User5	0,71	209875	1713
User6	0,53	82110	1181
User7	0,86	119140	1393
User8	0,87	102218	1311
User9	0,92	72672	1031
User10	0,51	51360	906
User11	0,80	145172	1454
User12	0,64	75437	1151
User13	0,81	121203	1389
User14	0,78	63187	1070
User15	0,84	105413	1277
User16	0,63	126422	1558
User17	0,66	87687	1209
User18	0,89	157140	1724
User19	0.82	121874	1295
User20	0,84	109457	1138

Table 6.2. stylus drawing results

User	Ratio	time (ms)	length (pixel)
User1	0,77	43171	985
User2	0,72	65703	958
User3	0,92	53937	971
User4	0,80	35735	1005
User5	0,65	61094	1027
User6	0,82	17438	934
User7	0,77	43171	985
User8	0,77	33656	973
User9	0,87	38031	1191
User10	0,77	43171	29141
User11	0,87	29141	968
User12	0,88	30016	970
User13	0,83	64860	988
User14	0,88	23891	900
User15	0,88	46969	994
User16	0,80	29438	929
User17	0,75	60031	1019
User18	0,90	68469	1004
User19	0,87	61456	989
User20	0,85	76926	986

Table 6.3. mouse drawing results

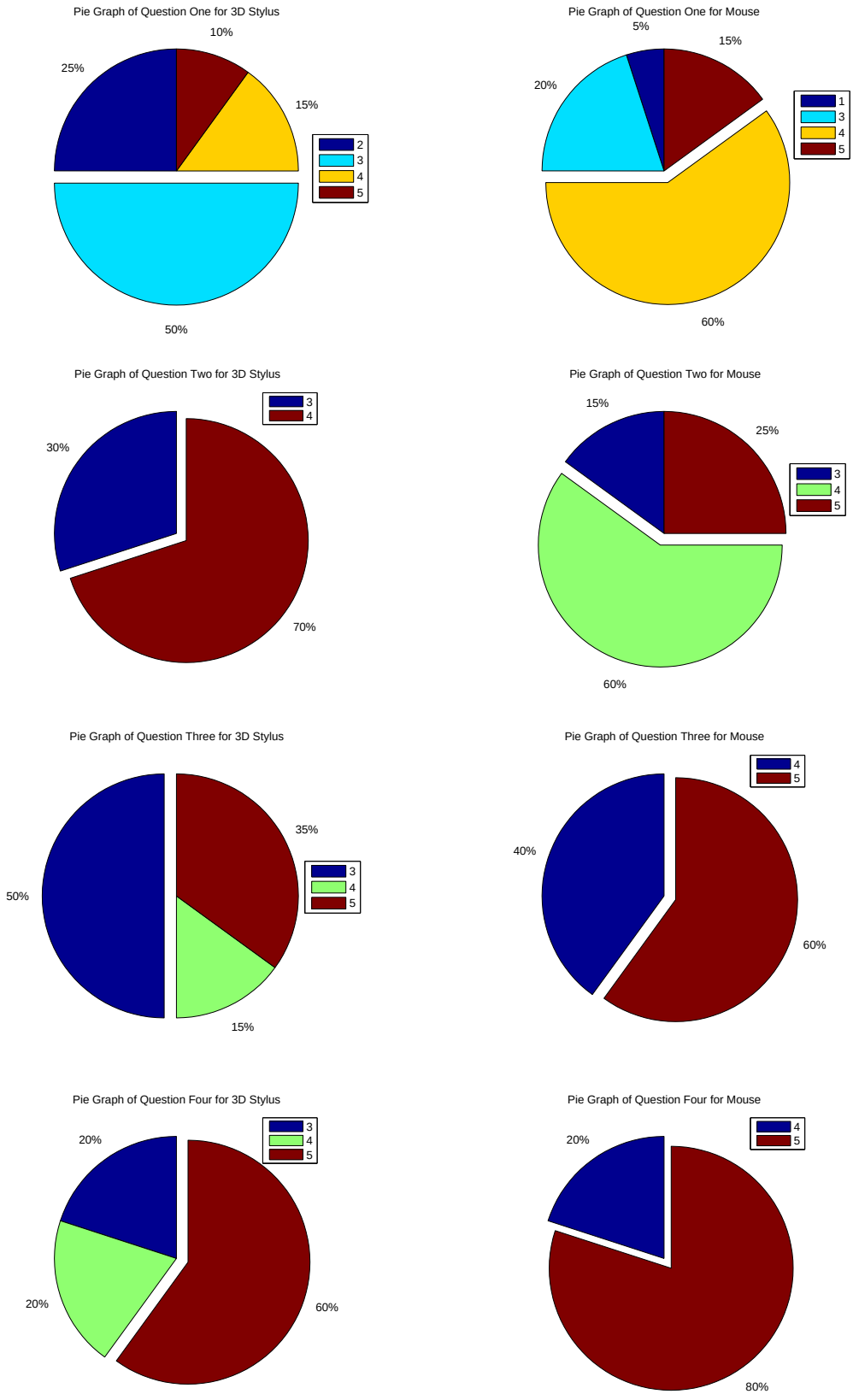

Figure 6.7. Orientation test screen. The white frame shows the limits of transparent cube. The white plane in the middle of the cube is slicing plane and it shows the cross-section of invisible volumetric letter

for drawing applications.

6.4. Discussion of Test Results

In this section we would like to discuss results of the tests separately because *3D Stylus* success is different for the two applications.

In the first test, *3D Stylus* was successful compare to mouse. The reason behind the success of *3D Stylus* originated from its innovative structure. In fact, considering all users are accustomed with mouse before, they are supposed to finish by mouse faster. Nevertheless mouse have 2 constraints which makes it slower.

The first thing make mouse slow is that changing the edges. The slicing plane turns around the axis which is defined by the two edges not across the picked edge. Therefore user always need to pick one edges after other and this move forth and back costs time. Sometimes the plane doesn't span the axis which user want to turn around. So, user firstly need to change orientation of plane to the axis which he wants to turn around.

The second problem of the mouse design could be seen in Figure 6.8. The user suffers from difficulty of picking edge of plane due to its small area caused by projective distortion. In this case user is supposed to turn the whole image. This is done by getting out of image and costs time. On the other hand, *3D Stylus* system also has some problems like getting out of fields of view. If the user stay far enough it is not mostly concern. It can be seen from the questionnaire that users didn't complain about getting stuck. On the other hand, the further away from the camera, the lower the resolution. In our case this is not so important but maybe for some very sensitive applications it could be a concern.

As can be seen from fitting line and comparison of mean time per letter using *3D Stylus* is at least twice more efficient than mouse. Questionnaire also support this evaluation.

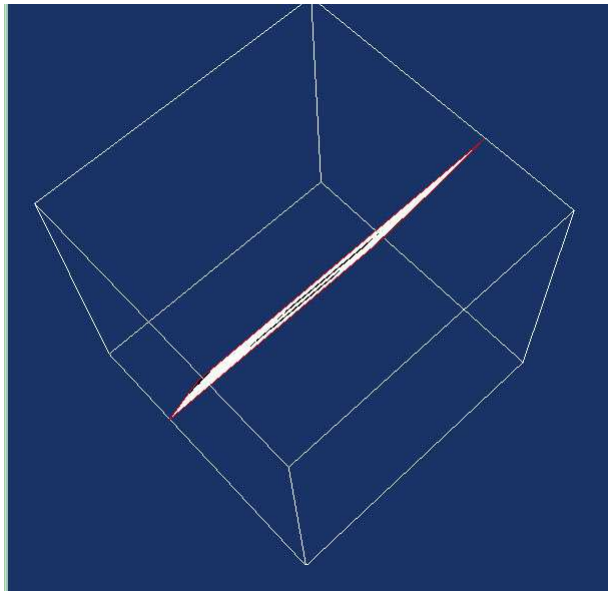

Figure 6.8. The picture illustrates the difficulty of picking edge to turn the slicing plane.

In the second test, it is understood that mouse is more efficient than *3D Stylus*. When we compare elapsed time, the drawing mouse time is nearly half of *3D Stylus* time. Also the ratio for mouse is higher.

Mouse is an interface developed for 2D applications and all users have been using it for years for this purpose. On the other hand, the *3D Stylus* is firstly using the design and have no experiences.

Second important difference is interpretation, some users complained that sometimes they expect the pointer go up while their hand go down. This is also very related with experiences. Because they know what happen while using mouse, there is no complain about mouse.

7. CONCLUSIONS

This thesis introduced a new human- computer interface using a passive object called "*3D Stylus*". System is based on stereo video processing. Design aimed at to be used in radiological rooms as a assistant tool for investigating 3D visualized patient data. User tests showed that results are hopeful.

3D Stylus is one of the convenient ways of oblique slicing. All test users reported advantage of *3D Stylus* during the orientation tests. On the other hand, we didn't found it efficient in 2D applications. Consequently, for 2D application mouse should be chosen due to its high resolution, intuitive 2D structure and habit.

Our design is working real time despite all stereo processing algorithms. For the future works system could be improved in terms of segmentation. However this improvement has the risk of exceeding processing time limits of a real time application.

Another possible improvement can be on filtering part. Both particle filter and kalman filter has different powerful sides. Therefore combining these two filters into one filter by cascading can give better results in terms of both error variance and mean error.

To sum up, *3D Stylus* will be one of the example HCIs developed for 3D environment. Using a stick for 3D application is shown that a good approach. Developing further application other than oblique slicing with the *3D Stylus* is very likely.

REFERENCES

1. Hachet, M., P. Guitton and P. Reuter, "The CAT for efficient 2D and 3D interaction as an alternative to mouse adaptations" *ACM Transaction On Graphics* 23 (3): 731-731, AUG 2004

2. *Wireless Data Glove: The CyberGlove II Wireless System*, http://www.immersion.com/3d/products/cyber_glove.php, 2007

3. *Welcome to the Centrum voor Wiskonde en Informatica*, http://www.cwi.nl, 2007

4. Grossman, T., D. Wigdor and R. Balakrishnan, "Multi-finger gestural interaction with 3d volumetric displays", *17th Annual ACM Symposium on User interface Software and Technology*, OCT 2004

5. Fleisch, T., G. Brunetti, P. Santos and A. Stork, "Stroke-Input Methods for Immersive Styling Environments", *Proceedings of the Shape Modeling International*, 2004

6. Piegl, L. and W. Tiller, *The NURBS Book*, Springer, ISBN-3-540-55069-0, pp. 405-453, 1995

7. Moustakas, K., D. Tzovaras, S. Carbini, O. Bernier, J. E. Viallet, S. Raidt, M. Mancas, M. Dimiccoli, E. Yagci, S. Balci, E. I. Leon, , "MASTER-PIECE: A Multimodal (Gesture+Speech) Interface for 3D Model Search and Retrieval Integrated in a Virtual Assembly Application", *Proceedings of ENTERFACE'05*, 2005

8. Betke, M., J. Gips and P. Fleming, "The camera mouse : visual tracking of bodu features to provide computer access for people with severe disabilities", *IEEE Transaction on Neural systems and rehabilitation engineering*, Vol.10, No. 1, pp. 1-10, MAR 2002

9. Hermann, T., T. Henning and H. Ritter "Gesture Desk -An integrated multi-modal gestural workplace for sonification", *Gesture-based communication in human-computer interaction lecture notes in artificial intelligence* Vol. 2915, pp. 369-379, 2003

10. Huang, C. L. and C. Y. Chung, "A real-time model-based human motion tracking and analysis for human computer interface systems", *EURASIP journal on applied signal processing*, Vol. 11, pp. 1648-1662, SEP 2004

11. Fukunaga, K., *Introduction to Statistical Pattern Recognition*, Elsevier, 1990

12. Gonzalez, R. and R. Woods, *Digital Image Processing*, Addison-Wesley Publishing Company, 2002

13. *Least-Squares Line*, http://www.efunda.com/math/leastsquares/lstsqr1dcurve.cfm, 2007

14. Kalman, R. E. and R. S. Bucy, "New Results in Linear Filtering and Prediction Theory", *Transactions of the ASME - Journal of Basic Engineering* Vol. 83, pp. 95-107, 1961

15. Kent, J. T., "The Fisher-Bingham distribution on the sphere", *J. Royal. Stat. Soc.*, vol.44, 71-80, 1982

16. Forstyh, D. and J. Ponce, *Computer Vision A Modern Approach*, Prentice Hall, 2003

17. *Kalman filter - Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/Kalman_filter, 2007

18. Julier, S. J. and J.K. Uhlmann, "A New Extension of the Kalman Filter to non-linear Systems" *In The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management II, SPIE*, 1997

19. Hartley, R. and A. Zisserman, *Multiple View Geometry*, Cambridge University Press, 2003

20. Isard, M. and A. Blake, "CONDENSATION- conditional density propogation for visual tracking", *International Journal of Computer Vision*, Vol.29:1, pp.5-28, 1998

21. Longuet-Higgins, H. C., "A computer algorithm for reconstructing a scene from two projections", *Nature*, Vol.293, pp.133135, 1980

22. *Point Grey Research Inc. -Home*, www.ptgrey.com, 2007

23. *Point Grey Research Inc. -Stereo Vision Products - Digiclops SDK*, http://www.ptgrey.com/products/digiclopsSDK/index.asp, 2007

24. *Point Grey Research Inc. -Stereo Vision Products - Triclops SDK*, http://www.ptgrey.com/products/triclopsSDK/index.asp, 2007

25. *Learn Visual C++ 2005*, http://msdn2.microsoft.com/en-us/visualc/aa336412.aspx, 2007

26. *Open CV library Overview - Open Source Computer Vision Library*, www.intel.com/technology/computing/opencv/overview.htm, 2007

27. *VTK Home Page*, www.vtk.org, 2007

28. *SourceForge.net: Welcome to SourceForge.net*, www.sourceforge.net, 2007