Ex.arb. 5518

CHALMERS

Wii Remote Based Head Tracking in 3D Audio Rendering

Master of Science Thesis

YASHAR DELDJOO

Division of Signal Processing
Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden, 2009
Report No. EX034/2009

Wii Remote Based Head Tracking in 3D Audio Rendering

Master of Science Thesis

Yashar Deldjoo
yashar.deldjoo@gmail.com

Supervisor: Dr. Erlendur Karlsson, Ericsson Research.
Examiner: Prof. Irene Y.H.Gu, Chalmers University of Technology

Abstract

The 3D audio scene rendered for a listener moving in a 3D environment depends directly on the listener's position and orientation in the three dimensional space. Tracking the listener's position and orientation is, therefore, an important component in interactive virtual environments.

By mounting four diodes on the listener's headset, a stationary Wii-Remote can be used to track the head movements of the listener (both position and orientation) and provide the 3D audio rendering motor with the necessary 6DOF data on the listener's movements to do a fully interactive rendering of the audio scene.

In this project, a head tracking solution is developed and is integrated into a 3D audio rendering engine to obtain an interactive virtual environment. Head pose estimation is the central part in the head tracking algorithm. Prior to tracking, 3D feature points are obtained and used as a model. Pose is then estimated by solving a robust version of "Perspective n-Point Problem". In order to analyze the accuracy of the algorithm, results were validated primarily by simulation of a random walk movement of a listener in MATLAB and used as the input to the head tracking algorithm. Thereafter, the solution was compiled as a library in C and tested in real time tracking. A graphical output is used to demonstrate the applicability of our method in real-time application. The final solution is integrated into a 3D audio rendering engine as a part of the virtual environment.

In relation to a closely related work by Johnny Chung Lee on Wii-Remote for head tracking (can be found in YouTube), this work tracks the full 6DOF movement of the listener (translation + orientation) while Chung lee's work only tracked the translation (3DOF) assuming that the orientation was the negated translation.

CHALMERS
HUVUDBIBLIOTEK
2011 -02- 25

Acknowledgments

Now it's time to thank each and every person without whom this work wouldn't have been possible. I would like to show my respect and deepest appreciation to my supervisor Erlendur Karlsson for continuously encouraging me with his patient guidance and friendly communication, throughout the whole period of time. Dr. Erlendur Karlsson, you are not just a professional, but an excellent researcher with broad vision. Million thanks for giving me the opportunity to work in the multimedia research group of Ericsson and for proposing such an interesting and challenging thesis work, despite your extremely busy schedule.

I am also grateful to Fredrik Jansson, the manager of audio group and all the members in the multimedia group for their kind help and contribution on such a nice working environment.

I would also like to thank my examiner Prof. Irene Gu at Chalmers for supporting me to be engaged in the thesis work and for her valuable help and suggestions.

Last but not least, I would like to thank all my friends and family, for their trust, continuous support and patience.

"Mum and Dad, now it's time to thank you although these words are not just enough. Despite the distance factor, your wisdom, advice and continuous support influenced my everyday decisions and were an advice to make what I am now. Arash, you always had and still have your own way of support. I am very lucky to have a brother like you and I owe you a big thank you. My deepest gratitude to Mary, Mohit and Moheb my dear cousins for their special hospitality and constant support and advice. You are my main source of motivation and inspiration pushing me always to the best. It's really hard to thank people like you. I wish you the best in your future plans. Finally, thanks to everyone who made me have a nice stay in Sweden. "

"The whole of science is nothing more than a refinement of everyday thinking."
Albert Einstein

Yashar Deldjoo
Nov 29th,2009

Table of Contents

Abbreviations

FOV	Field Of View
HRTF	Head Related Transfer Function
IID	Interaural Intensity Difference
ITD	Interaural Time Difference
VR	Virtual Reality
6DOF	Six Degree Of Freedom
3D	Three Dimensional

1 Introduction

1.1 3D audio rendering with head tracking

"Everyday life is full of three-dimensional sound experiences" [5] . Humans, in order to interact with their surroundings and make sense of the environment, need to obtain spatial awareness through their bodily senses *vision* and *hearing*. The human auditory system is a powerful perceptual tool that plays a major part in this matter. Through it, we are able to recognize the sound source location in our surroundings and to identify and follow the movement of moving sources.

But how can the human perform the sound source localisation task? The mechanism underlying directional sound perception is based on the listener having two ears or **binaural hearing**. The only signals available to our hearing are the signals picked up by our two ear drums and all the spatial information provided by our spatial hearing are deduced from those two signals.

The main acoustic cues used for localization are interaural differences in time **ITD**[1] and in intensity **IID**[2] of the eardrum signals. The sound waves approaching us from a given direction reach each ear with different intensity and arrival time as illustrated in figure 1. In addition, both ear signals are subject to complicated filtering process caused by interaction with the torso, head and in particular the external ear or pinna, before reaching the eardrums. These factors modify the frequency content of the signals by reinforcing some frequencies and attenuating others in a manner that they depend on the *direction* of the incident sound.

The brain uses the timing, intensity and spectral *differences* in the signals received by two ears to determine the location of the incident sound. These indicators are known as **spatial cues** [5][39].

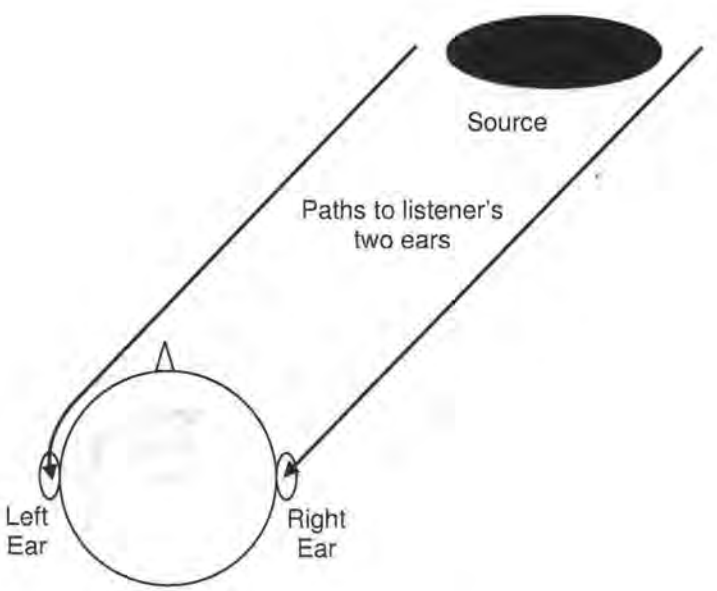


Figure 1 *A single sound source emitting a wave front that is received separately by two ears. Different paths to the listener's two ears result in different delays and intensities[39].*

[1] Interaural Time Difference
[2] Interaural Intensity Difference

The filters that define the transformation of a sound from a point in 3D space to the eardrums are known as **Head Related Transfer Functions** (HRTFs) and be can be estimated from measurements accurately. The head related transfer functions are complicated functions of frequency which are *unique* for every angle and position of incidence and contain all the tonal transformation of a particular sound source relative to the listener's head. Head related transfer functions are measured for many locations of the sound source relative to the head, resulting in a database of hundreds of HRTFs. HRTFs capture all the spatial cues for sound localization and it follows that sounds can be spatialized artificially using well-known digital filter algorithms, if the HRTF is known [10].

By the same token, a **Virtual 3D Audio** system imitates the process of natural hearing and simulates the wave propagation in a 3D environment that can trigger the spatial hearing of the listener. This is done by means of a pair of measured HRTFs as the digital audio filters. When a sound signal is processed by the digital filters and listened to over a headphone, the listener will experience the sense of actually being in a 3D environment and perceives sounds as coming from different directions around him, as specified by the HRTFs. This process is called **binaural synthesis** and is illustrated in figure 2:

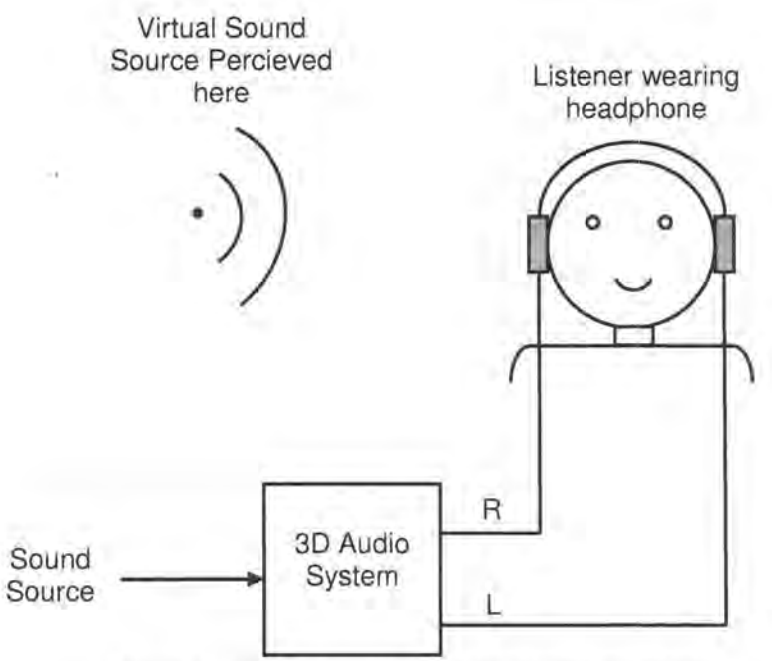


Figure 2 *3D audio system using binaural synthesis.*

However, with no information to the 3D audio renderer about the listener's *head movements*, the rendered scene will be relative to the listener and thus moves with the head movements of the listener, which is not as it should be in reality. The reason is that the virtual sound is simulated for a specific position of the sound source with respect to the listener and when the listener's location changes in 3D space, the 3D audio system still functions but then it assumes that the listener has a particular orientation. In order to have a **Dynamic Virtual Environment**, there is a need for a mechanism that provides the positional information of the listener to the 3D audio renderer to update the digital filters and produce binaural signals which are in full agreement with listener's location. This mechanism is known as **Head Tracking**.

Therefore, if one of the perceptual requirements is to have the virtual sound image remain fixed in virtual space independent of how the listener moves his head, a **Six-Degree Of Freedom Head Tracking** system is necessary and the user of the 3D audio system is the central point in a dynamic virtual environment [4]. Positional data about the listener's position and orientation in relationship to a reference coordinate system can be obtained in terms of a rotation specified by the rotational angles *pitch*, *head*, and *roll* and a translation specified by a translation vector $[t_x,t_y,t_z]^T$ as illustrated in figure 3:

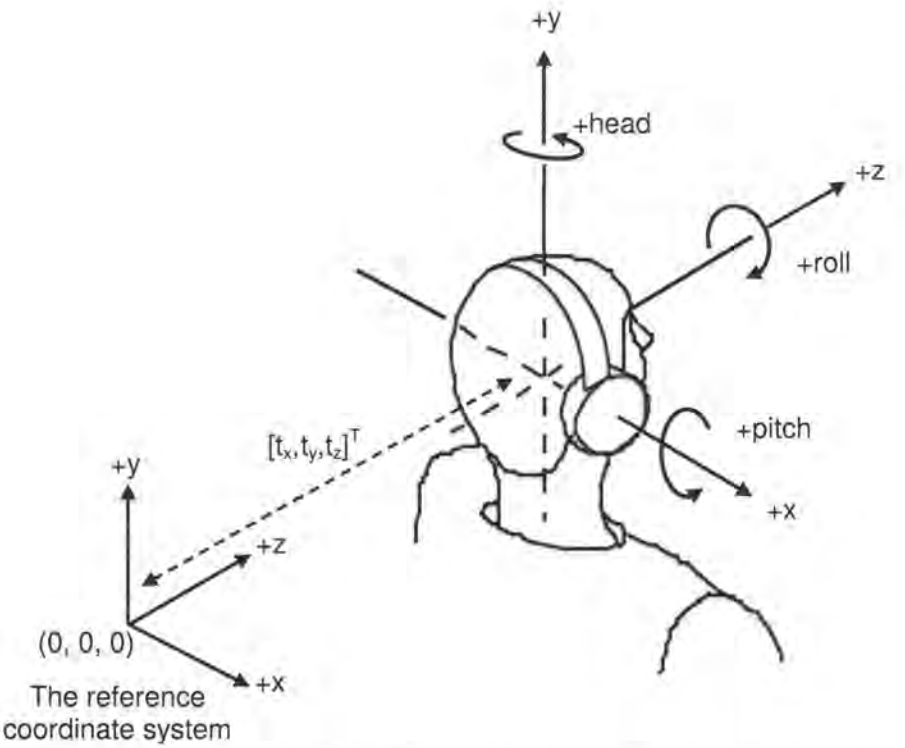


Figure 3 *The axis rotations pitch, head and roll and translation specified by a translation vector* $[t_x, t_y, t_z]^T$ *in a head tracking system [4].*

The head tracking problem can be solved by using a head tracking device that is attached to the listener's head. This device determines the listeners head position and orientation with respect to a reference point and sends this information to the computer to update the binaural signals associated with the head movements of the listener. The problem is that ordinary head tracking devices are very clumsy and not applicable in everyman's home [4] .

In this master's thesis project, we use the Nintendo **Wii Remote**[3] for the task of tracking. The Wii Remote represents a new class of sensor devices that was a revolution at the time it was presented due to the interactive gaming capabilities that it introduced. It is the primary controller which ships with Wii game console from Nintendo. What distinguishes the Wii Remote from traditional controllers is that it is equipped with a 3-axis linear accelerometer that can measure the acceleration that the device makes. This ability enables the user to interact with and manipulate the items in the screen just by moving the Wii Remote without touching buttons. Therefore natural movements can be simulated like the ones made in reality while playing games, like tennis or bowling [41] .

In addition to motion sensing capability, the Wii remote can track infrared light sources. It is equipped with a built-in optical IR tracker camera that can track up to four infrared light sources. By mounting some infrared LEDs on listener's head, we use the information seen by the camera and delivered by the Wii remote to estimate the listener's head position and orientation in 3D space. With a head tracking solution in place, the results are delivered to a 3D audio rendering engine to update the listener's orientation vectors. The result will be a correctly rendered 3D audio space.

[3] Sometimes referred to as Wiimote

1.2 The objective of master's thesis

1.2.1 Problem definition

The aim of this master's thesis is to implement a robust Wii remote based head tracking solution that enables us to perform full 6DOF[4] head tracking for the 3D audio rendering engine.The project is divided into two main problems:

1. Head tracking problem
2. Integration of the head tracking solution into the 3D audio system

1. Head tracking problem

The need for a solution arises from the fact that we need to obtain the head *position* and *orientation* of the listener in a Virtual 3D Audio Environment based on the feedbacks from the Wii Remote. The idea is to put Infrared light sources on a headset and track the listener's head movement by estimating the position and orientation of the listener from the data reported by the Wii remote camera. This is done by measuring the (x,y) coordinates (2D data) from the IR camera and translating them into 3-dimensinal coordinates. We contemplate having to use all four points available to us from the Wii remote to obtain position and orientation - six degrees of freedom 6DOF- of the listener's head in 3D space. The solution will be simulated and analyzed in MATLAB before all else. Subsequently it will be developed and complied as a library in C to be integrated into the 3D audio rendering engine in the next phase.

The Wii Remote uses a Bluetooth connection to communicate with the computer. The development of the head tracking solution will then entail getting the appropriate Bluetooth device drivers to work on a PC and interfacing the 3D audio rendering engine to the device drivers, and developing a head tracking solution based on the IR light positioning.

2. 3D audio integration problem

The next portion of this study consists of integrating the head tracking solution into a 3D audio rendering engine. The 3D audio system is equipped with all the functionalities that can trigger the spatial hearing of the listener. In order to be effective, the 3D audio system should be integrated into the head tracking library to obtain real-time information of the listener's position and orientation. With head tracking in place, the user's head will act as an input to the system and enables the 3D audio system to generate discernable directional sounds based on user's head location.

A Graphical User Interface (GUI) is also created to provide a mechanism to assess the applicability of the head tracking in a real time 3D audio rendering demo. The graphical user interface (GUI) has projection of 3D space in three 2D planes that can visualize the tracking performance in 3D space from 3 viewing positions. It acquires the movement data from real time measurement data and thus can model the behavior of the user's movement in 3D space.

[4] 3DOF Position + 3DOF Orientation

1.2.2 Solution specification

The solution to the problem should be able to:

1. Collect raw camera image plane data from the Wii remote
2. From the 2D camera data calculate the **head pose** that is position and orientation of the infrared rigid body mounted on the listener's headphone
3. Deliver the result of pose estimation to the 3D audio rendering engine

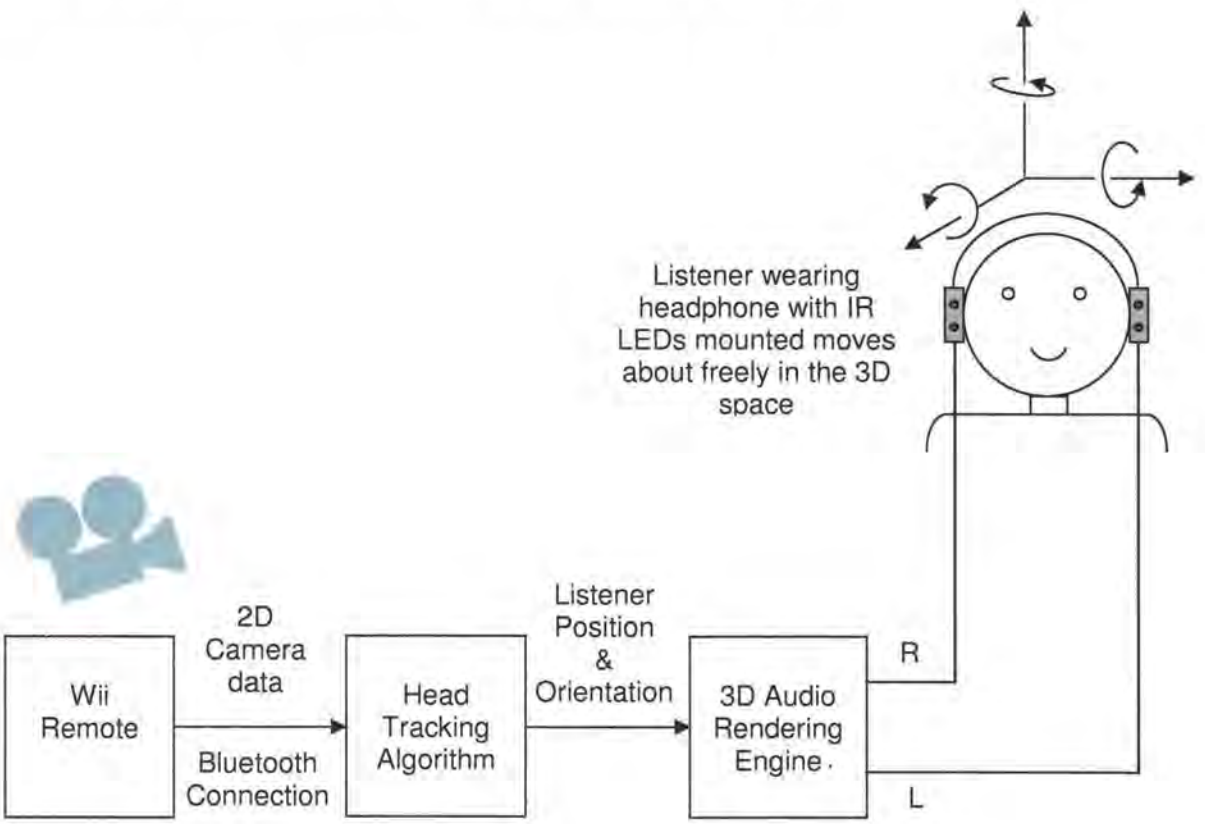


Figure 4: *The architecture of our head tracking algorithm.*

A listener is wearing a headphone with IR LEDs mounted and can move about freely in the 3D space. The Wii remote camera sees the IR diodes and reports the dots location in its image plane to the computer via Bluetooth communication. The head tracking algorithm calculates the head pose, given the image plane data. The results are supplied to the 3D audio rendering engine for updating binaural signals.

1.3 Similar works

Recently, a graduate student from Carnegie Mellon University Johnny Chung Lee created a Wii remote based head tracking system for virtual reality display that has received a lot of attention on Youtube. By using a head mounted sensor bar and placing the Wii remote at the base of the display, Lee is able to track the location of the head and render view dependent images on the display. The desktop VR display then reacts to head movement and becomes a portal into another space as if it were a real window. Lee recreates the similar effects of virtual reality interface with just some simple software and a low-cost Wii remote [20].

In essence, the motion of an object in three-dimensional space can be modeled with six parameters, three for rotation and three for translation. The movement along each of these six is independent of each other and therefore it is referred to as six degree of freedom (6DOF). Johnny Lee's head tracking solution however, assumes that the observer is always looking into the middle of the screen and estimates the translation based on this assumption. It cannot identify the head rotation as it should in reality. The math for 3DOF head tracking on is considerably simpler compared to that for 6DOF and is based on an approximation to compute the depth information and the other associated parameters.

Inspired by the work of John Lee, this system was defined for this project. The goal was to make the 3D Audio Rendering Engine applicable for a moving listener in a real time application, a reason for the choice of a head tracking system. The head tracking algorithm should calculate 6DOF pose of the moving head in the space and update these information for as many time as it is possible. The data available for this purpose, is exclusively four target points (and not more) which noticeably, restricts the choice of methodologies.

More closely related to this work is Oliver Kreylos' 6DOF input device where a single hand-held Wii remote is pointed towards a fixed tracking target. His approach to solve the system of equation associated with the pose estimation problem is based on a method known as *direct rigid body transformation*. For information can be found on [16],[17].

1.4 Different tracking approaches

Head tracking is a special case of *motion tracking or motion capture*. There has been a lot of research into motion tracking in the past, which started early in 1970s and 1980s for biomechanics research. In general, motion tracking refers to the process of detecting and tracking the motion of an object in 3D space. The information from the tracking could be used in *3D animation* to animate digital characters or for navigation in *virtual environments*.

Current tracking devices are based on optical and non-optical technologies as summarized in figure 5. This section studies these technologies and related systems together with a discussion of their advantages and disadvantages.

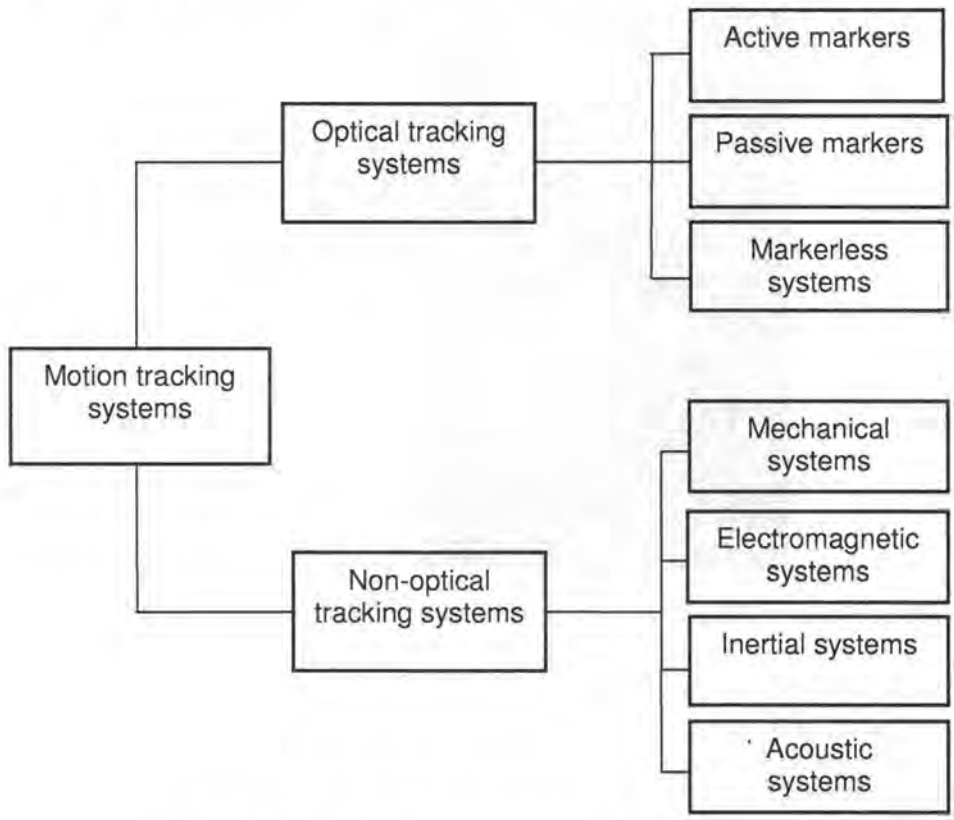


Figure 5 *Motion tracking systems and techniques*

1.4.1 Optical systems

Optical systems are used greatly in virtual environments for tracking head position and orientation. Optical systems rely upon measurements from *reflected* or *emitted* lights. They consist of two major components: light sources and optical sensors (cameras). They use data collected from optical sensors to track objects in space. This is done by measuring light intensity or processing images collected by the cameras.

The sensors are usually placed in a circle around the object. The optical tracking software combines the data from one or several cameras to calculate the position and orientation of the tracked object in the space. *Triangulation* can be used to combine the input of multiple cameras.

The light sources are known as **markers**. The markers are attached to the tracked object and send out lights that are recognized by the camera as bright spots in the captured images. There are two main variants of markers: *active markers* and *passive markers*.

In general, optical tracking systems have high update rates and sufficiently low latency. The disadvantage is that they suffer from line of sight (occlusion) problem. Also the environment light radiation can adversely affect the optical tracker performance. As result, the surrounding environment must be designed in a manner to eliminate as much as possible these causes of uncertainty[2], [26].

1.4.1.1 Active markers

Active markers **emit** light themselves. IR emitting light sources are usually chosen since they are invisible for human and does not interfere with regular light sources. Figure 6 shows an example of active markers. The image on the left is photograph of a subject during the capture session and the one on the right shows the reported marker position when picture was taken [18]:

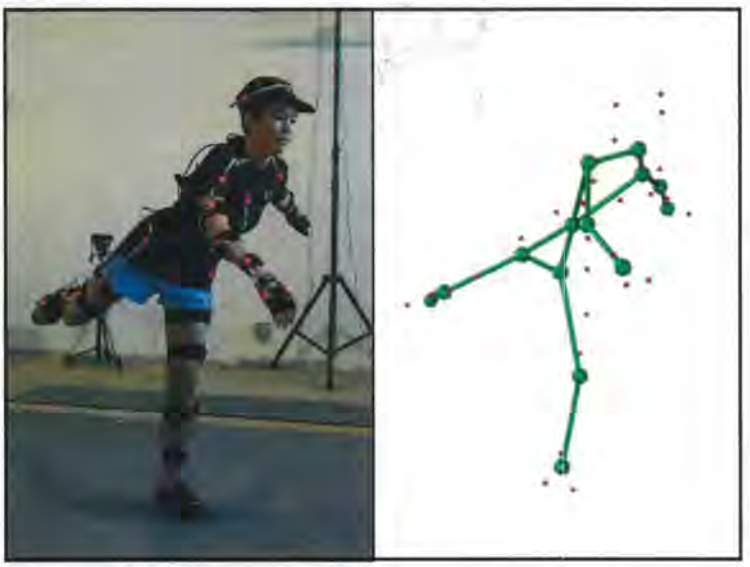


Figure 6 *The use of active markers in motion tracking [18]*

One advantage of using active markers is that they have higher radiation power (compared with passive markers) and thus can be recognized from further distances. The other advantage is that it is possible to flash each marker in a particular manner so that it can help the tracking software to identify markers. This can be greatly important in real-time systems. Their disadvantage, however, is that they need to be supplied by a power source either by attaching a source to the user or connecting him with wires. This may limit the subject in his freedom of movement.

1.4.1.2 Passive markers

Passive markers do not emit light themselves rather they **reflect** lights originating from light sources around the object. Therefore these types of markers should be coated with a retroreflective material than can reflect light back to its source with minimum scattering of light. This way many markers can be used and the subject does not need to carry any electrical equipments. The disadvantage of passive markers is that it is difficult to distinguish them in the image captured by the camera. The knowledge of relative distance of the markers as well as the use of multiple cameras can help with identifying the markers. Figure 7 shows an example motion tracking system with passive markers.

Figure 7 *Several passive markers are placed at specific points on subject's body for motion capturing*[5].

1.4.1.3 Markerless systems

Markerless systems do not require the user to wear any special equipment for tracking. They utilize *computer vision* techniques to estimate the actor's position and orientation.

1.4.2 Non-optical systems

Beside optical systems there are other tracking technologies which return position/and or orientation in 3D space. These systems may use one of mechanical, electromagnetic, inertial and acoustic technologies.

1.4.2.1 Mechanical systems

Mechanical systems measure position and orientation by using a direct mechanical connection between the tracked target and a reference point (environment). The actor has to attach a skeletal-like structure like an articulated arm to his body which has sensors mounted on the joint of the 'bones'. As he moves, the mechanical parts move as well. The sensors on the joints sense the motion and measure the change in position and orientation with respect to the reference point.

Mechanical systems are conceptually very simple approaches and cheap. They are precise and accurate. They have high update rate and low latency. The main disadvantage is that the user's motion is constrained by the mechanical arm. An example of mechanical system is shown in figure 8, which is a mechanical tracking device developed by Fake Space Lab [2],[26] .

[5] Picture from http://tyrell-innovations-usa.com/

Figure 8 *A mechanical tracking device developed by Fake Space lab*[6].

1.4.2.2 Electromagnetic systems

Electromagnetic tracking devices function by measuring the strength of magnetic fields. The magnetic fields are generated by three orthogonal coils on the transmitter side. On each sensor also three coils are embedded in a small unit that is attached to the tracked object. The measurements result on nine numbers representing the strength of each field in three directions which are used to measure the position and orientation of the sensor relative to the source.

The advantage over optical system is that electromagnetic systems don't have any occlusion (line of sight) issues. The other advantage is that one source may excite many sensors hence many objects can be tracked from a single source. The disadvantage, however, is that they have limited range (typically 1 to 3m) and may experience interference problem while operating in vicinity of metal objects[2],[26].

1.4.2.3 Inertial systems

Inertial systems utilize inertial sensors such as accelerometers to measure acceleration of the object. The motion data are usually sent to a computer over a wireless connection. Most inertial systems use gyroscopes[7] to measure rotational changes. If a six-degree of freedom tracking is required, the system should be combined with some position tracking devices.

The advantage is that they are cheap and easily affordable. Also, another advantage is that the subject has an infinite freedom of movement without any occlusion issues. The disadvantage is the drift between the actual and reported values that is accumulated over time[2],[26].

1.4.2.4 Acoustic systems

Acoustic systems rely upon transmission and sensing of sound waves for measuring the position and orientation of the target object. The waves are ultrasonic[8] sound waves which function on a frequency greater than the upper limit of human hearing. Higher frequencies are beneficial to avoid interference. Also most noise sources fall off with increasing frequency.

---

[6] Picture from http://www.informatik.umu.se/~jwworth/2Techniques
[7] A device for measuring orientation
[8] High frequency

Figure 9 *Intersense IS-900 Receiver and Transmitter components (Inertial and Ultrasonic tracking)* [9]

There are two ways of ultrasonic measurements: the so called *time-of-flight tracking* and *phase-coherence tracking.*

The first one is based measuring the flight duration of brief ultrasonic pulses. The transmitter emits sounds at given time slots. By measuring the time it takes the sound waves to reach the sensors located at fixed positions, the tracking software can calculate the distance from target to each sensor and the position of the target can be determined. For finding position, one transmitter suffices. For orientation, three sensors are required. By doing the same calculation for each sensor and finding the difference in location, orientation can be determined.

The second method is based on finding the phase difference between the sound waves emitted by the transmitter on the target and the ones emitted from a reference point. The phase of a wave carries the information about its position and thus the system can track the position of the target. Similar to time-of-flight tracking, by using multiple transmitters, orientation can be calculated. The phase-coherence tracking systems are subject to error accumulation problem since they are not based on measuring absolute position rather on periodic updates of position.

Radio and microwave tracking works in a similar way.

The disadvantage of acoustic systems is uncertainty in speed of sound due to physical properties of environment such as temperature and humidity. They are also very susceptible to acoustic noise and echoes in the physical environment.

More detailed information on tracking technologies can be found in [2] , [26].

[9] Picture from http://vislab.isums.edu/VIS/flex.html

1.5 Applications of head-tracked 3D audio

1.5.1 Virtual environments

A *Virtual Environment* is a computer–simulated program that allows the user interaction in a manner that gives him the impression of being in a real world or imaginary world. Virtual environments come in two main *visual* and *auditory* types.

In a virtual audio environment, a key issue is the degree to which we are able to track human interaction. For this reason, head tracked virtual audio system represent the best approach to implementing sound in a dynamic environment so that sound sources remain fixed in virtual space independent of head movement as they are in natural spatial hearing.

A generic example is computer workstation 3D audio with all types of sonic input. Consider a headphone wearing work station user who has directionalized all the sonic inputs to a specific location around him. He uses an audio spatial mapping to correspond to a *prioritizing* scheme. For telephone calls for example, spatialization becomes interesting when the callers can be identified.

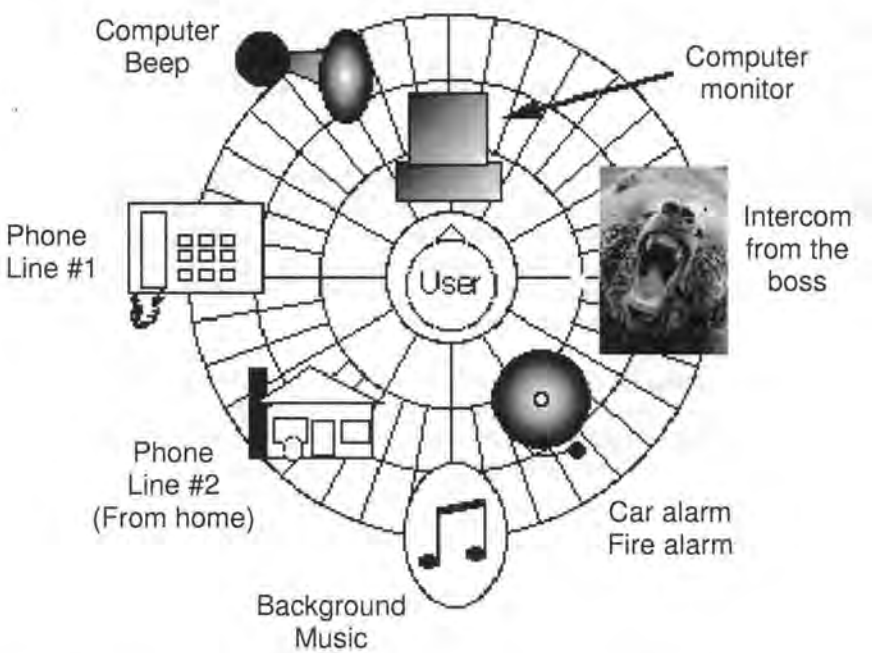


Figure 10 *Layout for a hypothetical GUI for arranging a set of 3D sounds.*

For instance, subordinate calls can be signaled by a phone ringing from rear and a call from home could always ring from front right; and the intercom from the boss from center and front. In such a scenario, in a prioritizing manner spatial location informs the listener whether or not to put someone on hold or activate a phone mail system. Similarly, other types of inputs that could be spatialized are such as building fire alarms, signals for sonification of a computer processing in another room as well as background music [4] .

1.5.2 Telepresence

When a person has the experience of being in a location which he or she is not actually in through the use of a set of technologies, it is called *telepresence*. The idea behind telepresence is that the person experiencing it should feel fully submerged with different parties at remote locations, being able to interact with the environment and feeling as though he or she is actually there. The interaction between parties can be made more effective if the technology uses as much *perceptual* information as possible.

The technology supports the development of large visual displays or 3D displays for visual communication. With interactive support of head movement, the auditory displays could be considered to add a desirable enhancement to telecommunication interfaces, facilitating the reproduction of voices of the parties in such a manner so they are perceived as coming from particular locations in space.

When communicating with people in such an environment, our perception of the scene is both *auditory* and *visual* and is inherently spatial and thus gives you the feeling of being in the other location [4],[9].

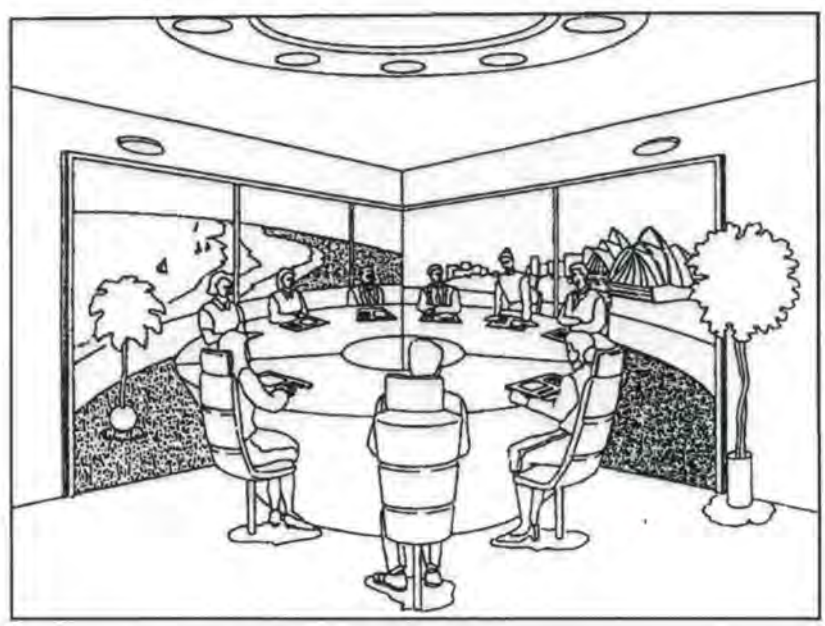


Figure 11 *A future telepresence system*

2 Background

In order to understand how a head tracking algorithm is designed, it is important to understand the underlying mathematics and geometrical concepts.

A common terminology used in this area is *rigid body*. By definition rigid body is a solid (stiff) body that the forces acting upon it do not change the relative position of its parts. A rigid body can undergo a transformation called *rigid body transformation* and the transformation changes the location and orientation of the object but not its shape that is to say the geometric model of the object stays unreformed. A rigid body in space has six degree of freedom (6DOF), three for location (X, Y, Z) and three for its orientation (pitch, roll, head) with respect to a fixed coordinate system. Deformable objects such as the human face or articulated objects like human bodies require more parameters for a precise description of their form. Examples of rigid body transformations are translation and rotation.

A major issue associated with head tracking is the *pose estimation problem*. For rigid bodies, pose estimation refers to describing an object in six degree of freedom. The pose estimation of the rigid body is estimated by finding the model parameters, translation and rotation, that once projected into the image plane of the camera, are in best agreement with the captured 2D data. The orientation and position of the camera in the object space are called the *camera extrinsic parameters*. Finding the 6DOF pose with respect to camera is the equivalent to finding the camera's extrinsic parameters.

Pose Estimation problem could be solved in different ways. This section studies two of the common methods, known as *the Perspective N-Point problem* and *Direct Rigid Body Transformation* which have a different problem formulation and use their own system of nonlinear equations.

2.1 Pose estimation

2.1.1 Definition

Pose Estimation is an important operation in many areas ranging from robotics to computer vision. The information from pose estimation can be used in object manipulation and self-localization of mobile robots or object tracking in audio-video tracking systems.

By definition, pose estimation consists of estimating the position and orientation of a known 3D object with respect to the reference coordinate system of the camera used to track the object. In other words, we search for a transformation of the 3D reference object such that the transformed object corresponds to 2D image data. For rigid objects, such transformation can be defined in terms of relative Rotation (R) and Translation (T) between two 3D objects or what is referred to as the *Euclidean Transformation*.

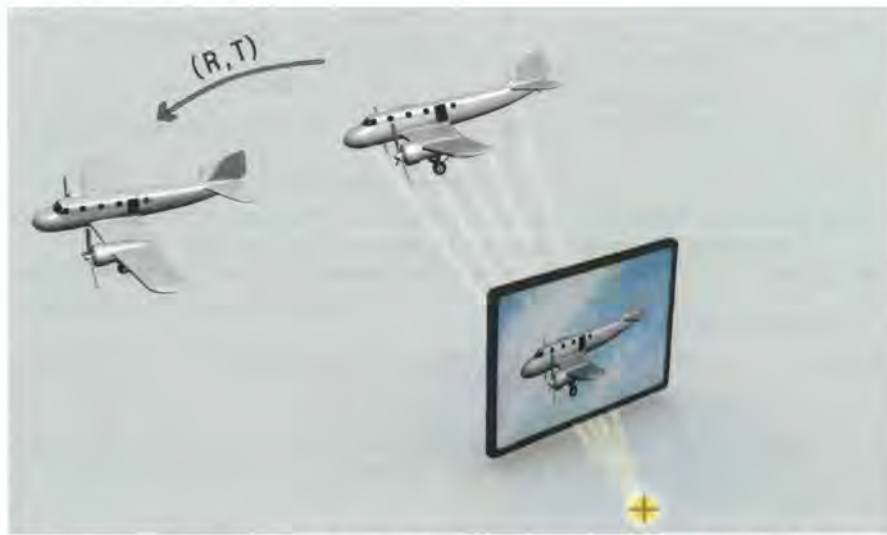


Figure 12 *Pose estimation in terms of Rotation (R) and Translation (T) of the rigid body from the 2D camera image plane data.*

Figure 13 models the behaviour of a camera recording in reality. In real world, we deal with a reference 3D rigid body which is rotated and translated to give the transformed rigid body, or the actual rigid body which is seen by the camera. The camera recognizes the rigid body points and projects them on its image plane. These data are 2D data and are obtained from a mapping process from 3D to 2D defined by the camera projection equations.

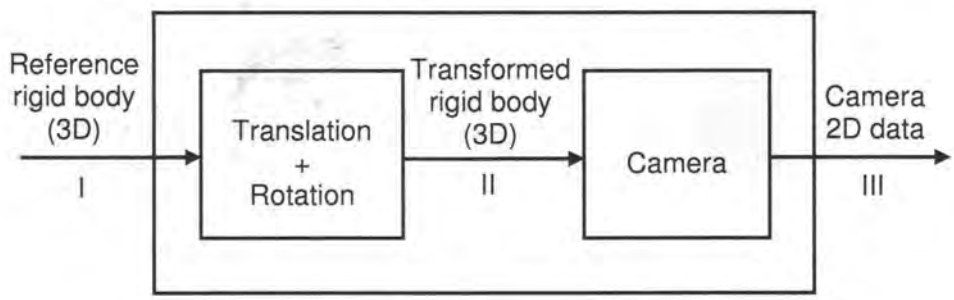


Figure 13 *Model of a camera recording in reality*

Correspondingly, in this chapter we explain each of the processes involved to obtain camera image plane data which are:

- Translation and rotation
- Projection of the 3D data to camera's 2D image plane defined by the pinhole camera model

On the other hand, pose estimation follows the process shown in figure 13 in a reverse manner, meaning from given 2D camera image plane data (III) and by knowing the geometry of the reference rigid body (I), it estimates the translation and rotation of the transformed rigid body (II) with respect to a reference rigid body (I). This process is shown in figure 14:

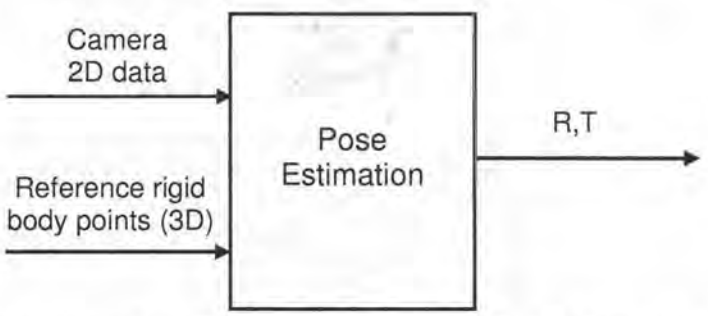


Figure 14 *Pose estimation means to estimate rotation and translation of a transformed rigid body from the 2D camera data and a known reference rigid body.*

2.1.2 Translation and rotation

Translation

Translation is an operation that displaces a point from one location to another by a fixed distance in a given direction. This is equivalent to changing the origin of coordinate system. Translation is represented as below:

$$T_{t_x,t_y,t_z}(x,y,z) = \begin{bmatrix} x+t_x & y+t_y & z+t_z \end{bmatrix}^T \quad (1)$$

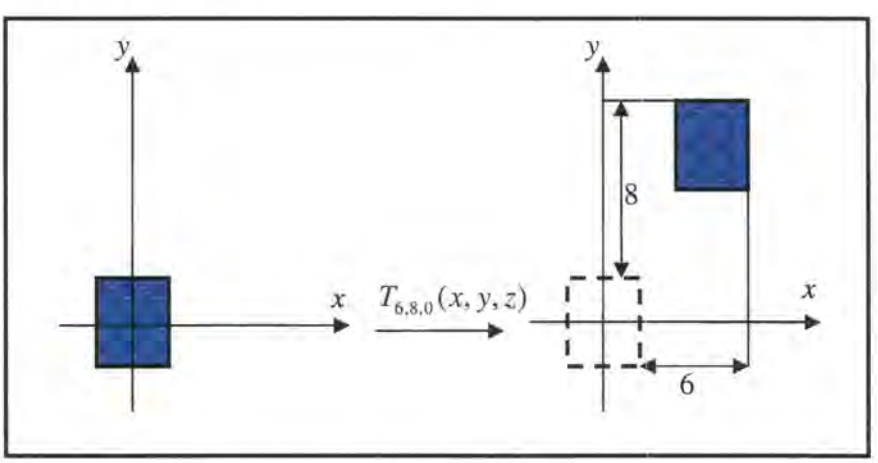


Figure 15 *Transformation with a translation $T_{6,8,0}(x,y,z)$ whereby the shape is moved 6 to right and 8 upwards*

Rotation

Rotation is a more elaborate set of transforms, because more parameters are involved in rotation matrices.

This type of transformation incorporates three rotation matrices $R_x(p), R_y(h), R_z(r)$ given by equations: [10]

$$R_x(p) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(p) & -\sin(p) \\ 0 & \sin(p) & \cos(p) \end{pmatrix}$$
$$R_y(h) = \begin{pmatrix} \cos(h) & 0 & \sin(h) \\ 0 & 1 & 0 \\ -\sin(h) & 0 & \cos(h) \end{pmatrix} \quad (2)$$
$$R_z(r) = \begin{pmatrix} \cos(r) & -\sin(r) & 0 \\ \sin(r) & \cos(r) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

These matrices represent counter clockwise rotation of an object relative to a fixed coordinate axis, by the angle ϕ ,around the x, y and z axes.

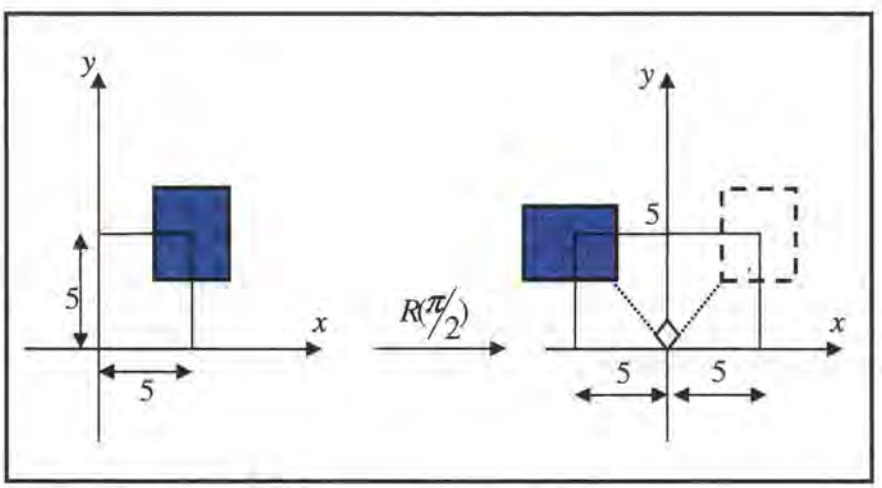


Figure 16 *90 degrees rotation in two dimensions*

Rotation matrices can be obtained using matrix multiplication. The order of this multiplication is important. The matrix product can be obtained is known as **Euler's Transform** and can be obtained as below:

[10] The rotation matrices operate on column vectors as Rv where R is a matrix of size $N \times N$ and v a vector with size $N \times 1$ (N=2 or 3).

$$E(h,p,r) = R_z(r)R_x(p)R_y(h) =$$

$$\begin{pmatrix} \cos(r)\cos(h) - \sin(r)\sin(p)\sin(h) & -\sin(r)\cos(p) & \cos(r)\sin(h) + \sin(r)\sin(p)\cos(h) \\ \sin(r)\cos(h) + \cos(r)\sin(p)\sin(h) & \cos(r)\cos(p) & \sin(r)\sin(h) - \cos(r)\sin(p)\cos(h) \\ -\cos(p)\sin(h) & \sin(p) & \cos(p)\cos(h) \end{pmatrix} \quad (3)$$

Euler's Transform is the matrix multiplication of three matrices namely as ***Pitch***, ***Head*** and ***Roll***. The angles h, p and r represent in which order and how much the head, pitch and roll should rotate around their respective axes.

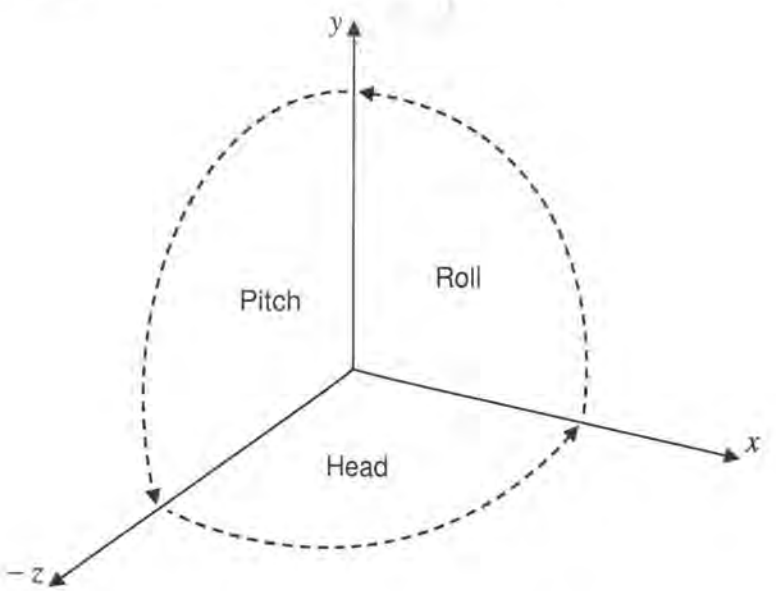


Figure 17 *The convention for Pitch, Roll and Head angles, Euler Angles*

These angles could be associated with different types of head movements. For example, changing the **head** angle makes the person *shake* his head "no", changing the **pitch** makes him *nod*, and **rolling** makes him tilt his head *sideways*.

Properties of rotation matrix:

All discussions below are for a 3×3 rotation matrix R.

a) *The **columns** of the rotation matrix R represent the rotated unit basis vectors of the original space.*

In the unrotated space, the unit basis vectors $\boldsymbol{a}_x, \boldsymbol{a}_y, \boldsymbol{a}_z$ lie along the basis vectors of the coordinate system:

$$\boldsymbol{a}_x = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \boldsymbol{a}_y = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \boldsymbol{a}_z = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{4}$$

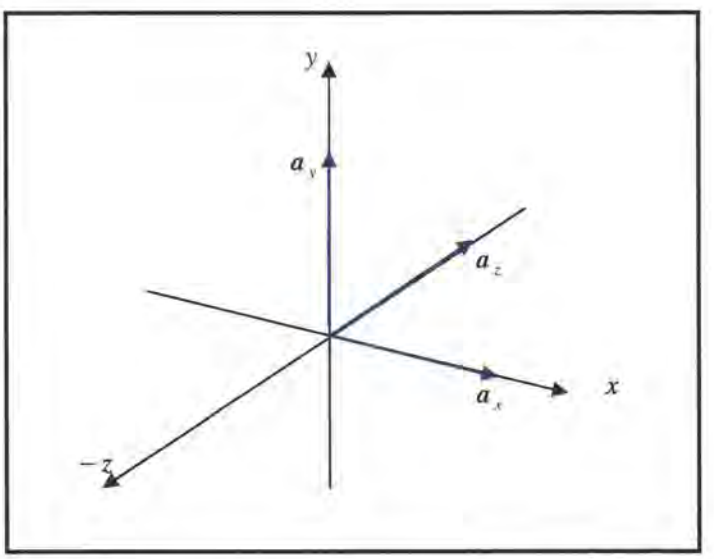


Figure 18 Orientation vectors in the unrotated space corresponds to the columns of the rotation matrix

In matrix notation, the transformation of the vectors could be represented in terms of matrix multiplication:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \boldsymbol{R} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{5}$$

where

$$\boldsymbol{R} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \tag{6}$$

If we apply the rotation the matrix to the unit basis vectors, the rotated vectors will correspond to columns of the rotation matrix.

$$\boldsymbol{a}_x' = \boldsymbol{R}\boldsymbol{a}_x = \begin{pmatrix} R_{11} \\ R_{21} \\ R_{31} \end{pmatrix} \quad \boldsymbol{a}_y' = \boldsymbol{R}\boldsymbol{a}_y = \begin{pmatrix} R_{12} \\ R_{22} \\ R_{32} \end{pmatrix} \quad \boldsymbol{a}_z' = \boldsymbol{R}\boldsymbol{a}_z = \begin{pmatrix} R_{13} \\ R_{23} \\ R_{33} \end{pmatrix} \tag{7}$$

$$\rightarrow \boldsymbol{R} = \begin{pmatrix} a_x' & a_y' & a_z' \end{pmatrix}$$

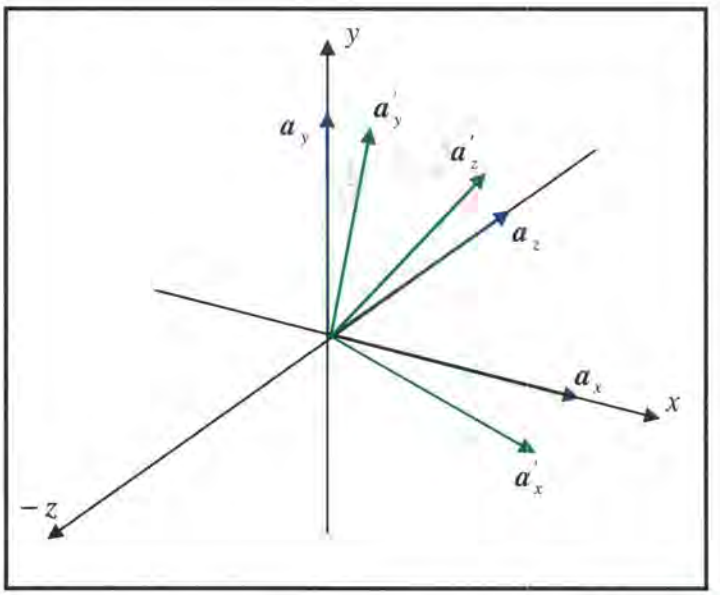


Figure 19 *Unrotated unit basis vectors (**blue**), the rotated vectors (**green**)*

The knowledge of this property is so useful because it can help us to find the rotated orientation vectors directly from the rotation matrix.

Example: Let us consider a simple example in 2D for a 60-degree rotation:

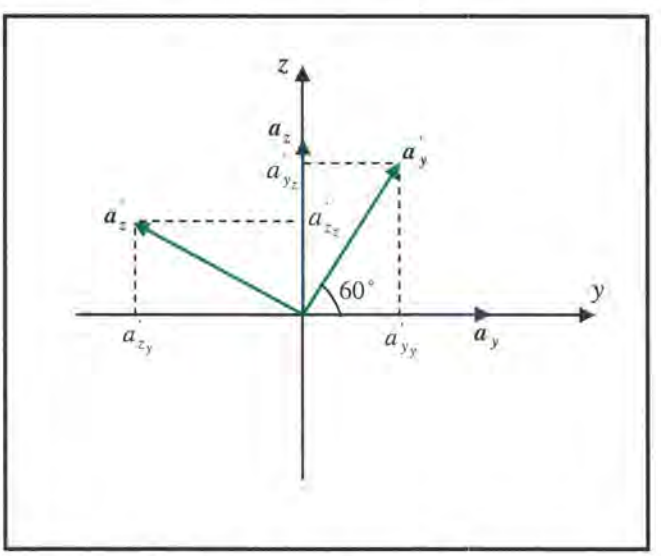


Figure 20 *60 degrees rotation in 2D*

The vectors are all of size one so we can find the rotated unit basis vectors as below:

$$
\begin{aligned}
a_{y_1} &= \cos(60) = \frac{1}{2}, a_{y_2} = \sin(60) = \frac{\sqrt{3}}{2} \\
a_{z_1} &= -\sin(60) = \frac{-\sqrt{3}}{2}, a_{z_2} = \cos(60) = \frac{1}{2} \\
\boldsymbol{a}_y &= \begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{pmatrix}, \boldsymbol{a}_z = \begin{pmatrix} \frac{-\sqrt{3}}{2} \\ \frac{1}{2} \end{pmatrix}
\end{aligned} \tag{8}
$$

The corresponding rotation matrix is defined as below as below in which the columns of the rotation matrix represent the rotated unit basis vectors.

$$
\phi = 60^\circ, \boldsymbol{R} = \boldsymbol{R}_x(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{a}_y & \boldsymbol{a}_z \end{pmatrix} \tag{9}
$$

*b) The **rows** of the rotation matrix R represent the coordinates in the rotated space with respect to the original space.*

Consider two reference coordinate systems F_1 and F_2, and a vector $\mathbf{v}$ whose components are known in F_1, represented as $\{\mathbf{V}\}_1$

$$
\{\mathbf{V}\}_1 = \begin{pmatrix} V_{x1} \\ V_{y1} \\ V_{z1} \end{pmatrix} \tag{10}
$$

We wish to determine the representation of the same vector in F_2, or $\{\mathbf{V}\}_2$:

$$
\{\mathbf{V}\}_2 = \begin{pmatrix} V_{x2} \\ V_{y2} \\ V_{z2} \end{pmatrix} \tag{11}
$$

The fixed vector $\mathbf{v}$ could be represented as linear combination of the basis vectors in two coordinate systems. The basis vectors are assumed to be orthonormal and they span the whole 3D space:

$$
\mathbf{V} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_{x1} \\ V_{y1} \\ V_{z1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_3 \end{pmatrix} \begin{pmatrix} V_{x2} \\ V_{y2} \\ V_{z2} \end{pmatrix} = \boldsymbol{B} \cdot \begin{pmatrix} V_{x2} \\ V_{y2} \\ V_{z2} \end{pmatrix} \tag{12}
$$

where $\boldsymbol{b}_i = [b_{xi} \quad b_{yi} \quad b_{zi}]^T$ s are the orthonormal basis vectors in transformed coordinate system

$$
< \boldsymbol{b}_i, \boldsymbol{b}_j > = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \tag{13}
$$

We would like to know how the transformed coordinate system is expressed in terms of original space. These are linear spaces and the corresponding system of equation is a standard system of linear equation and the solution could be expressed in matrix notation as below:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} V_{x1} \\ V_{y1} \\ V_{z1} \end{pmatrix} = \boldsymbol{B}.\begin{pmatrix} V_{x2} \\ V_{y2} \\ V_{z2} \end{pmatrix} \rightarrow \begin{pmatrix} V_{x2} \\ V_{y2} \\ V_{z2} \end{pmatrix} = \boldsymbol{B}^{-1}.\begin{pmatrix} V_{x1} \\ V_{y1} \\ V_{z1} \end{pmatrix} \tag{14}$$

And since the basis vector are orthonormal

$$\boldsymbol{B}^{-1} = \boldsymbol{B}^{T} \tag{15}$$

and therefore,

$$\boldsymbol{B}^{-1} = (\boldsymbol{b}_1 \quad \boldsymbol{b}_2 \quad \boldsymbol{b}_3)^T = \begin{pmatrix} \boldsymbol{b}_1^T \\ \boldsymbol{b}_2^T \\ \boldsymbol{b}_3^T \end{pmatrix} = \begin{pmatrix} b_{x1} & b_{y1} & b_{z1} \\ b_{x2} & b_{y2} & b_{z2} \\ b_{x3} & b_{y3} & b_{z3} \end{pmatrix} \tag{16}$$

Hence, the columns of new transformation matrix $\boldsymbol{B}^{-1}$ define the orientation vectors. Now, back to our earlier discussion on properties of rotation matrix, assuming that the transformation matrix is a rotation matrix:

$$\boldsymbol{R} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}$$

$$\boldsymbol{B} = \boldsymbol{R} \tag{17}$$

$$\begin{pmatrix} V_{x2} \\ V_{y2} \\ V_{z2} \end{pmatrix} = \boldsymbol{R}^{-1}.\begin{pmatrix} V_{x1} \\ V_{y1} \\ V_{z1} \end{pmatrix} = \begin{pmatrix} R_{11} & R_{21} & R_{31} \\ R_{12} & R_{22} & R_{32} \\ R_{13} & R_{23} & R_{33} \end{pmatrix}.\begin{pmatrix} V_{x1} \\ V_{y1} \\ V_{z1} \end{pmatrix}$$

Example (revisited):

Let us reconsider the previous example for 60-degree rotation in 2D. We can represent the rotation of the vectors by rotation of the coordinate system in opposite direction:

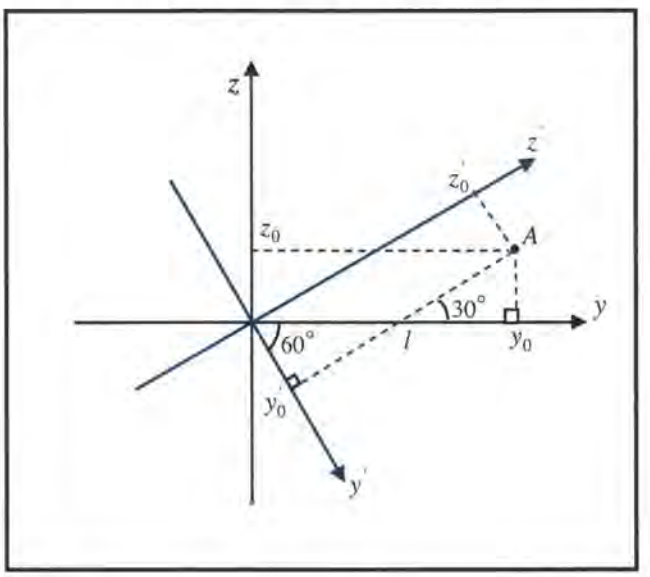


Figure 21 *60 degrees rotation in 2D corresponds to -60 degrees rotation of the coordinate system*

The coordinates in the rotated space could be evaluated as below:

$$\begin{aligned}\tan(30) &= \frac{z_0}{y_0 - l} \rightarrow y_0 - l = (\sqrt{3})z_0 \rightarrow l = y_0 - (\sqrt{3})z_0 \\ \cos(60) &= \frac{y_0'}{l} \rightarrow y_0' = \frac{1}{2}(y_0 - \sqrt{3}z_0) \rightarrow y_0' = \frac{1}{2}y_0 - \frac{\sqrt{3}}{2}z_0 = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}\begin{bmatrix} y_0 \\ z_0 \end{bmatrix}\end{aligned} \tag{18}$$

Doing the same calculations for z_0' we get:

$$z_0' = \frac{\sqrt{3}}{2}y_0 + \frac{1}{2}z_0 = \begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} y_0 \\ z_0 \end{bmatrix} \tag{19}$$

If we compare these coefficients with the corresponding rotation matrix:

$$\boldsymbol{R} = \begin{pmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix} \tag{20}$$

We realize that the above relations could be expression in terms of row elements of the rotation matrix:

$$\begin{bmatrix} y_0' \\ z_0' \end{bmatrix} = \begin{bmatrix} R_{row1} \\ R_{row2} \end{bmatrix}\begin{bmatrix} y_0 \\ z_0 \end{bmatrix} \tag{21}$$

c) *The rotation matrix is an **orthonormal** matrix.*

By definition, an orthonormal matrix is a special orthogonal matrix that has unit vector size. The two conditions are then orthogonality and unit vector size.

Orthogonality:

A matrix whose inverse is equal to its transpose is called an **orthogonal matrix**.

$$RR^T = I \rightarrow R^{-1} = R^T \tag{22}$$

where R^T is the transpose of R and I is the identity matrix. The other interpretation is that the dot product of any pair of rows or any pair of columns is 0.

$$\begin{cases} row_i \cdot row_j = 0 & if \quad i \neq j \\ col_i \cdot col_j = 0 & if \quad i = j \end{cases} \tag{23}$$

Unit Vector size:

This means the sum of squares of the elements in any row or column is equal to 1:

$$\begin{cases} row_i \cdot row_j = 1 & if \quad i \neq j \\ col_i \cdot col_j = 1 & if \quad i = j \end{cases} \tag{24}$$

2.2 The pinhole camera model

The *pinhole camera model or perspective camera model* is the simplest optical system for mathematical modelling of the projection of 3D points onto the image plane of a camera. This an idealized model and does not accurately consider some of the optical effects which are present in real cameras; however it's sufficient for many image processing and vision applications.

According to this model, the ideal pinhole camera is modelled as a box with a small hole on the front and a photographic plane at the opposite side as figure below:

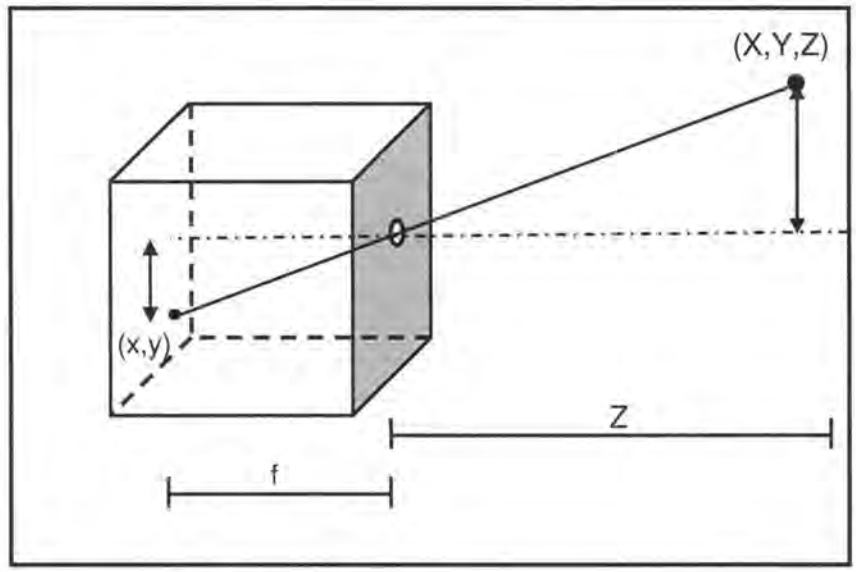


Figure 22 *The Pinhole Camera*

The pinhole model defines the projective geometry related to the imaging of points in 3D space. For instance, the camera's coordinate system is defined with the origin located at the centre of the projection, the so called focal point. The centre of projection is located at focal-length distance at the positive side of z-axis which is pointing toward the camera's *optical axis*. In this representation, focal length, *f*, is the distance from the focal point to the image plane. Figure 23 illustrates this:

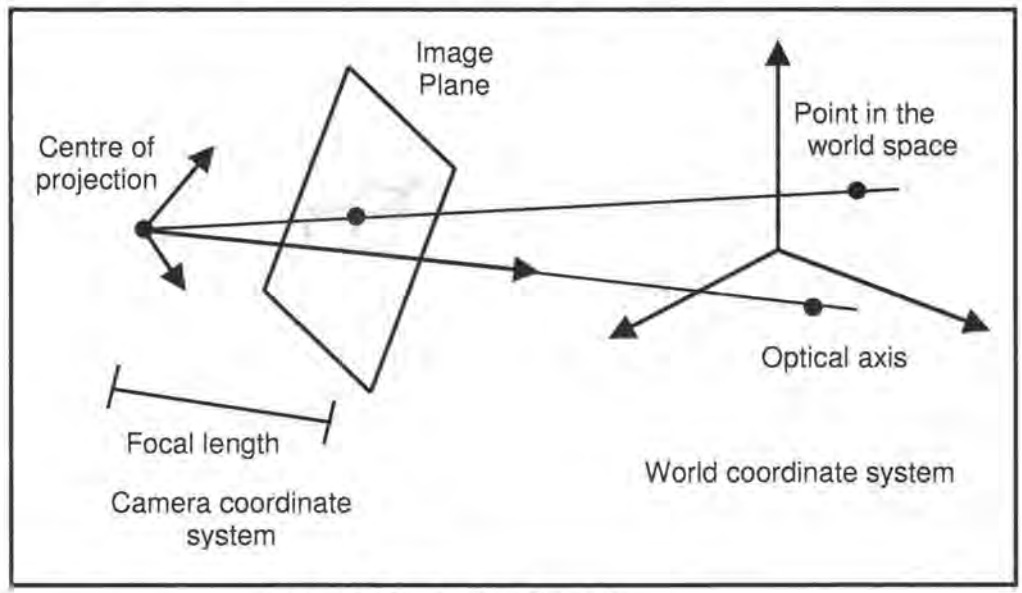


Figure 23 *The Pinhole Camera*

We now turn to the geometric aspect of image formation in a bit more detail. The aim is to link the space points with that of their corresponding image points. The notation used is such that we can distinguish between 2D and 3D points and the augmented points in homogenous coordinate systems.

Imagine, a 3D point *M* on an object with coordinates M=[X, Y, Z]^T in the *camera's coordinate system* as shown in figure 22. The image of this point denoted by m=[x, y]^T is formed by projection of the optical from the point to the centre of projection *C*. The intersection of this ray with the image plane forms the image projection point.

The reference frame, called the camera frame has significant importance in vision tasks. In the camera frame, the relationship between the 3D point *M* and its image projection *m* is formed by similar triangles and is given by:

$$\frac{x}{X}=\frac{f}{-Z}\rightarrow x=-f\frac{X}{Z}$$
$$\frac{y}{Y}=\frac{f}{-Z}\rightarrow y=-f\frac{Y}{Z} \qquad (25)$$

2.2.1 Camera extrinsic and intrinsic parameters

The reference camera coordinate system is introduced in order to define the fundamental equations of perspective projection in a simple form. However, the camera reference coordinate system (camera frame) is not always known and a common problem is determining the location and orientation of the camera frame with respect to some reference frame using only image information.

The *extrinsic parameters* or *external parameters* define any set of geometric parameters that forms uniquely the transformation between the unknown camera reference frame and the world reference frame. A typical choice for defining the transformation parameters between two frames is to use rotation matrix and translation vector.

1. **Translation** $\quad \boldsymbol{T}=[T_x,T_y,T_z]^T$
2. **Rotation** $\quad \boldsymbol{R}=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$

We should emphasize the outset that **although the rotation matrix consists of 9 parameters, it only has 3 independent rotation parameters or three degrees of freedom**. The translation vector has 3 degrees of freedom so totally there are 6 extrinsic parameters. These parameters change during tracking.

The *intrinsic parameters* or *internal parameters* are the camera's truly parameters that remain unchanged during tracking. These parameters define the optical, geometric and digital characteristics of the viewing camera. For a pinhole camera, they include:

- **Focal Length (*f*)** : The orthogonal distance from the optical centre camera focus point to the image plane in some physical units
- **Pixel Size (p_x,p_y)**: The dimension (width, height) of each pixel in physical units
- **Centre of the projection**: The intersection of the optical axis with the image plane

Further information can be found in [24] & [43].

2.2.2 Pose estimation methodologies

Determining the pose of a 3D object from the 2D projection of the 3D object onto the image plane of the camera and a reference model is an important segment of many problems in computer vision.

The pose estimation methodologies aim at estimating the rotation and translation of the rigid body from the 2D image plane data as shown below.

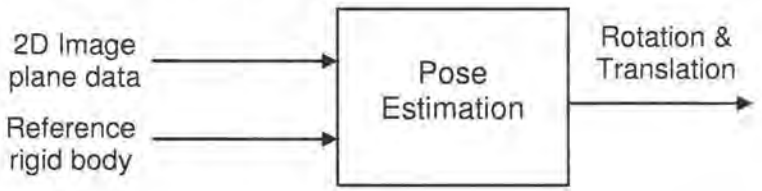


Figure 24 *Pose Estimation*

In this section, we present two methods for estimating the 3D pose from the image plane projections.

The first method is a two step process as shown and described below:

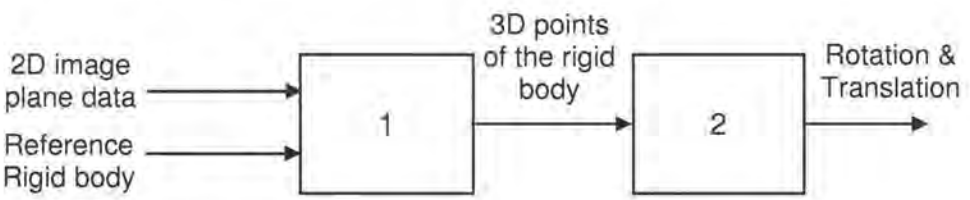


Figure 25 *Two-step Pose Estimation*

- In the first step, the 3D positions of the selected points of the rigid body are estimated from the image plane data.
- In second step, from the estimated 3D points of the rigid body, the rotation and translation of the rigid body is estimated.

The second method however, is a one-step process method which is based on finding the position and orientation of the rigid body directly. In other words, the equations for this method are formulated in manner that when solved, all unknown variables are evaluated at once and there is no need to go through multiple steps.

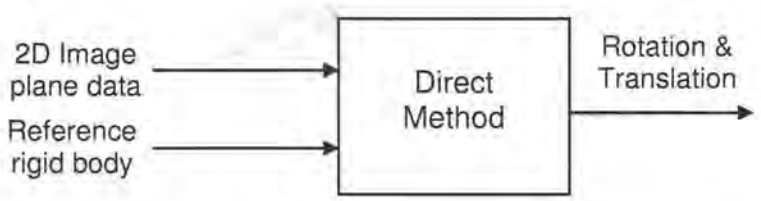


Figure 26 *Direct method for pose estimation*

Note:

All the methods discussed in the following sections are for $n = 3$ or $n = 4$ points, because the Wii remote can track up to maximum 4 simultaneous IR light points. Therefore, the study of more robust methods based on $n > 4$ points is off topic for this project.

2.2.2.1 Estimating the 3D points of the rigid body from the 2D image plane projection

Let us assume that we are given a set of n 3D points $P_i = [X_i, Y_i, Z_i]^T, i = 1,2,..,n$ where $[X_i, Y_i, Z_i]$ s are the unknown coordinates in the camera coordinate system. The 2D image locations of the observed points $q_i = [x_i, y_i]^T, i = 1,2,..,n$ are also known and reported by the camera:

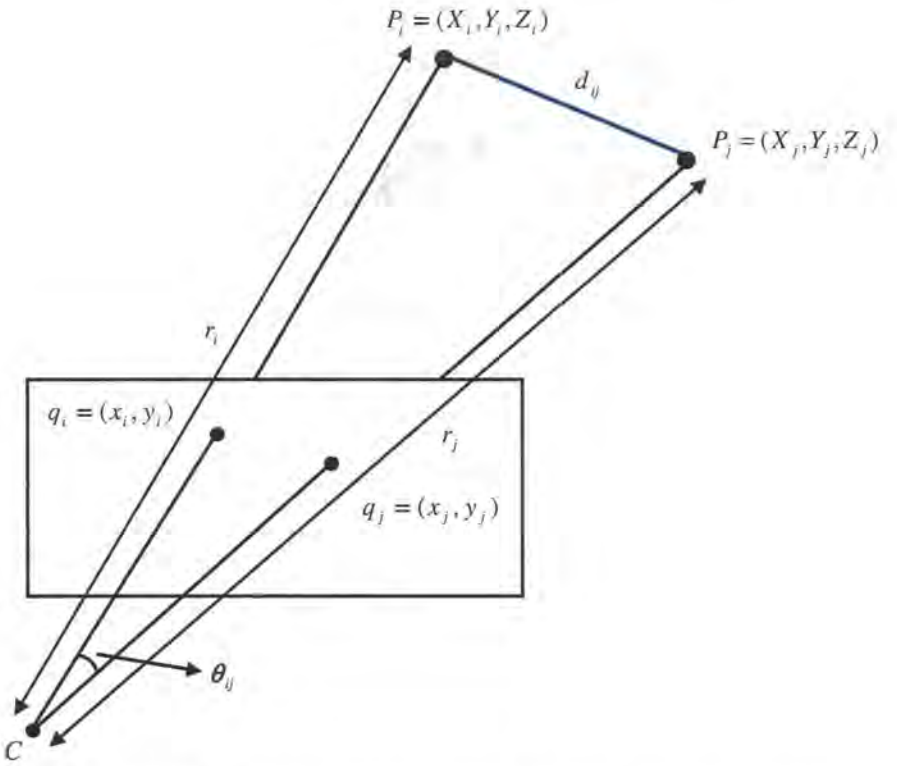


Figure 27 *Estimate 3D positions of the points from the 2D image plane data*

We wish to find the unknown 3D points $P_i = [X_i, Y_i, Z_i]^T, i = 1,2,..,n$.

1. The Perspective N-Point method:

According to figure 27, each pair of correspondence $P_i \leftrightarrow p_i$ and $P_j \leftrightarrow q_j$ gives a constraint on the unknown camera-point distances r_i and r_j according to the *law of cosines* as the following equation:

$$r_i^2 + r_j^2 - r_i r_j \cos\theta_{ij} = d_{ij}^2 \qquad (26)$$

where d_{ij} is the inter-point distance between the *i*th and *j*th points which is **known** to us as a *reference model* and θ_{ij} is the 3D viewing angle subtended at the camera centre by the *i*th and *j*th points.

The cosine of viewing angle is directly **evaluated** from the image plane data:

$$\cos\theta_{ij} = \frac{< q_i, q_j >}{|q_i||q_j|} \tag{27}$$

Equation is a quadratic formula with 2 unknowns. When 3 features (q_1, q_2, q_3) are available, the cosine relation can be reformulated for pairs $(q_1, q_2), (q_1, q_3), (q_2, q_3)$ and the system of expended equations will have three quadratic equations with three unknowns:

$$\begin{cases} r_1^2 + r_2^2 - r_1 r_2 \cos\theta_{12} = d_{12}^2 \\ r_1^2 + r_3^2 - r_1 r_3 \cos\theta_{13} = d_{13}^2 \\ r_2^2 + r_3^2 - r_2 r_3 \cos\theta_{23} = d_{23}^2 \end{cases} \tag{28}$$

The problem of finding the radians from the system of equation is referred to as Perspective 3-Point Problem (P3P) or PnP for any n-points.

$$z_0 = \frac{\sqrt{3}}{2} y_0 + \frac{1}{2} z_0 = \begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} y_0 \\ z_0 \end{bmatrix} \tag{29}$$

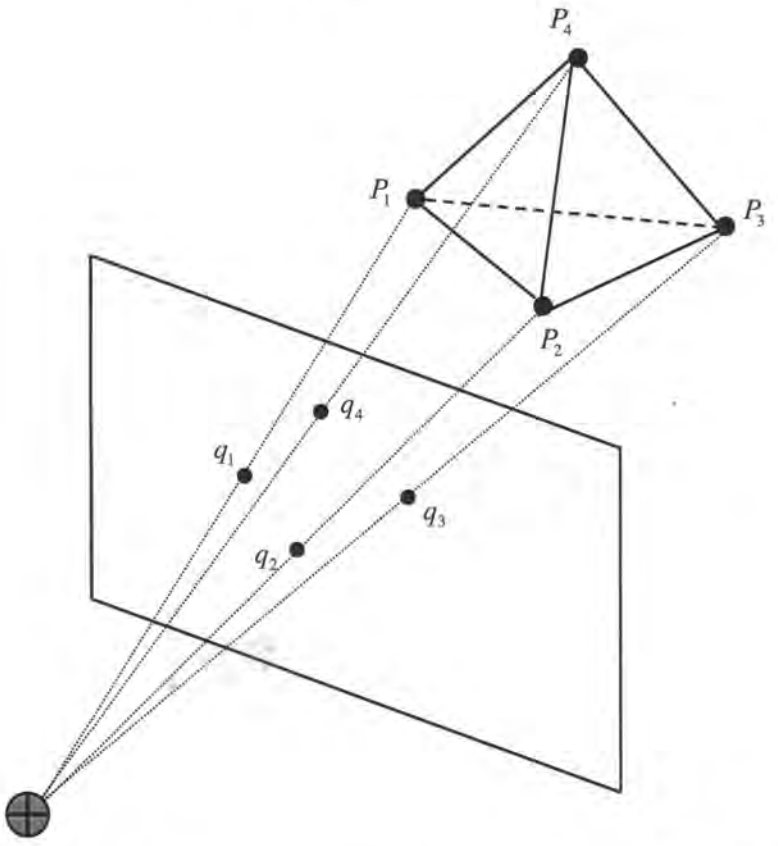


Figure 28 *Projections of four reference points on the camera's image plane-PnP method for n=4, P4P*

Solution to PnP method

The full system of non-linear equations can be solved using a variety of methods. In this section we introduce some of these methods:

- **Analytic solution:**

There is a well-known closed-form solution for PnP method. Several types of analytic methods are proposed in order to reformulate set of equations with several unknowns into a single equation with one unknown. To illustrate this, let us now consider the PnP method when n=3.

The following polynomial system could be obtained:

$$g_{ij}(r_i, r_j) = r_i^2 + r_j^2 - r_i r_j \cos\theta_{ij} - d_{ij}^2 = 0$$

$$\begin{cases} g_{12}(r_1, r_2) = 0 \\ g_{13}(r_1, r_3) = 0 \\ g_{23}(r_2, r_3) = 0 \end{cases} \tag{30}$$

According to Bezout bound theorem for polynomials, see appendix A, this system has a bound of $8 = 2 \times 2 \times 2$ solutions. Using Sylvester's resultant, one independent variable r_i is eliminated between two equations and further eliminations result in a corresponding eighth degree polynomial $O(n^8)$:

$$h(r_i) = a_5 r_i^8 + a_4 r_i^6 + a_3 r_i^4 + a_2 r_i^2 + a_1 \tag{31}$$

Since the characteristic equation has only even terms of r_i, the equation can be rewritten as a fourth degree polynomial in $x = r_i^2$:

$$h(x_i) = a_5 x_i^4 + a_4 x_i^3 + a_3 x_i^2 + a_2 x_i^1 + a_1 \tag{32}$$

The solutions for this equation match to that for original polynomial sets. This equation has at most **four** solutions which could be solved directly for x_i and only one of these solutions is of interest to us. Therefore we need to define a method to choose the desired answer from the solution subset. A more robust case of PnP method happens with n=4 points where we obtain an over-determined system of equation with 6 equations and 4 unknowns:

$$\begin{cases} g_{12}(r_1, r_2) = 0 \\ g_{13}(r_1, r_3) = 0 \\ g_{14}(r_1, r_4) = 0 \\ g_{23}(r_2, r_3) = 0 \\ g_{24}(r_2, r_4) = 0 \\ g_{34}(r_3, r_4) = 0 \end{cases} \tag{33}$$

Generally speaking, for n points, there exists $\frac{n(n-1)}{2}$ constraint of type $g_{ij}(r_i, r_j) = 0$ on the unknown distances $r_1, r_2, ..., r_n$ and correspondingly $\frac{(n-1)(n-2)}{2}$ quadratic polynomial of type $h(x_i) = 0$ in one variable $x_i = r_i^2$.

For n=4, three forth degree polynomial looks as follows:

$$\begin{cases} h_1(x) = a_{15}x^4 + a_{14}x^3 + a_{13}x^2 + a_{12}x + a_{11} = 0 \\ h_2(x) = a_{25}x^4 + a_{24}x^3 + a_{23}x^2 + a_{22}x + a_{21} = 0 \\ h_3(x) = a_{35}x^4 + a_{34}x^3 + a_{33}x^2 + a_{32}x + a_{31} = 0 \end{cases} \quad (34)$$

which can be written in matrix form

$$\begin{pmatrix} a_{15} & a_{14} & a_{13} & a_{12} & a_{11} \\ a_{25} & a_{24} & a_{23} & a_{22} & a_{21} \\ a_{35} & a_{34} & a_{33} & a_{32} & a_{31} \end{pmatrix} \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \end{pmatrix} = A_{3\times5} t_{5\times1} = 0 \quad (35)$$

There is a unique solution to the above system of equation which is an advantage of P4P method over P3P. The only requirement is that the points do not lie in some critical configuration such as a coplanar configuration. However, even if this situation happens in some cases, the method could be modified to a version of linear 4-point algorithm that handles the co-planarity cases.

For detailed information to derivations and solutions, the reader is referred to reference [34]. Therefore, *the Camera Pose is uniquely determined for all* $n \geq 4$ *points*. The higher number of points entered in the equations, the more accurate are the results.

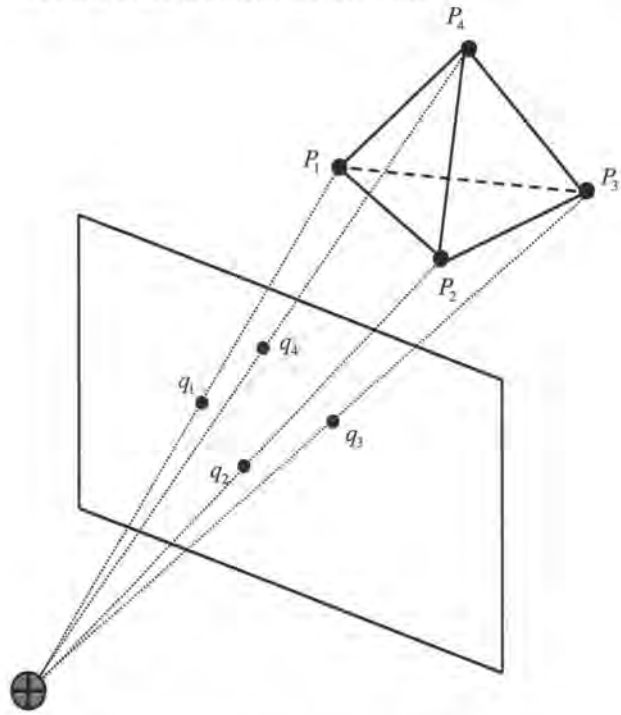


Figure 29 *Projections of four reference points on the camera's image plane-PnP method for n=4,P4P.*

- **Iterative solution:**

Projection from 3D to 2D is a non-linear operation and each of the methodologies for 6DOF pose estimation defines its own system of non-linear equation. The set of equations used in Perspective N-point Problem is a bilinear system of equation which needs to be solved uniquely.

This problem is a candidate for the application of Iterative based methods. The basic idea of *Iterative methods* or *optimization algorithms* is given an estimate of unknown camera-point distance, to evaluate the residual distance $(f_i(\boldsymbol{r}) - y_i)$ and try to minimize it iteratively:

$$
\begin{aligned}
& f_i(\boldsymbol{r}) = r_j^2 + r_k^2 - r_j r_k \cos\theta_{jk} \rightarrow \min \sum_{i=1}^{N} \| f_i(\boldsymbol{r}) - y_i \|^2 \\
& y_i = d_{jk}^2 \\
& \boldsymbol{r} = [r_1 \quad r_2 \quad r_3 \quad r_4]^T
\end{aligned}
\tag{36}
$$

In this definition, the PnP problem is recasted as a minimization problem that could be solved using one of the variant of the optimization methods such as *Gauss-Newton* or *Levenberg-Marquardt* algorithm.

2.2.2.2 Estimating the rotation and translation directly

1. Direct rigid body transformation

Given *n* 3D points on a rigid body $P_i = (X_i, Y_i, Z_i)$ and their image plane projections $q_i = (x_i, y_i)$, it is possible to uniquely recover absolute position and orientation of the rigid body. Rigid body transformation refers to the process of rotating (R) and Translating (T) the acquired model, to align with the recovered model.

The acquired model is a configured and priori known rigid body specified with their 3D points $P_i^w = (X_i^w, Y_i^w, Z_i^w)$ for $, i = 1,2,..,n$ which serves as the model rigid body and the recovered model is the rigid body recognized by the camera $P_i = (X_i, Y_i, Z_i), i = 1,2,..,n$ in each frame shown in figure 30:

The relationship between the 3D points of two rigid bodies is given by:

$$\begin{aligned} P_i &= \boldsymbol{R} P_i^w + \boldsymbol{T} \\ \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} &= \boldsymbol{R} \begin{bmatrix} X_i^w \\ Y_i^w \\ Z_i^w \end{bmatrix} + \boldsymbol{T} \end{aligned} \tag{37}$$

where $\boldsymbol{R}$ is the 3×3 rotation matrix

$$\boldsymbol{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{38}$$

and $\boldsymbol{T}$ is the 3×1 translation vector

$$\boldsymbol{T} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} \tag{39}$$

According to [45] "the rigid body transformation from one coordinate system (X_i^w, Y_i^w, Z_i^w) to another (X_i, Y_i, Z_i) is *unique* if the transformation is defined as a 3D rotation around the origin followed by the 3D translation."

We can rewrite the (37) equations as:

$$\begin{aligned} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} &= \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \begin{bmatrix} X_i^w \\ Y_i^w \\ Z_i^w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \\ \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} &= \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} P_i^w + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \end{aligned} \tag{40}$$

where r_i s are the row elements of the rotation matrix.

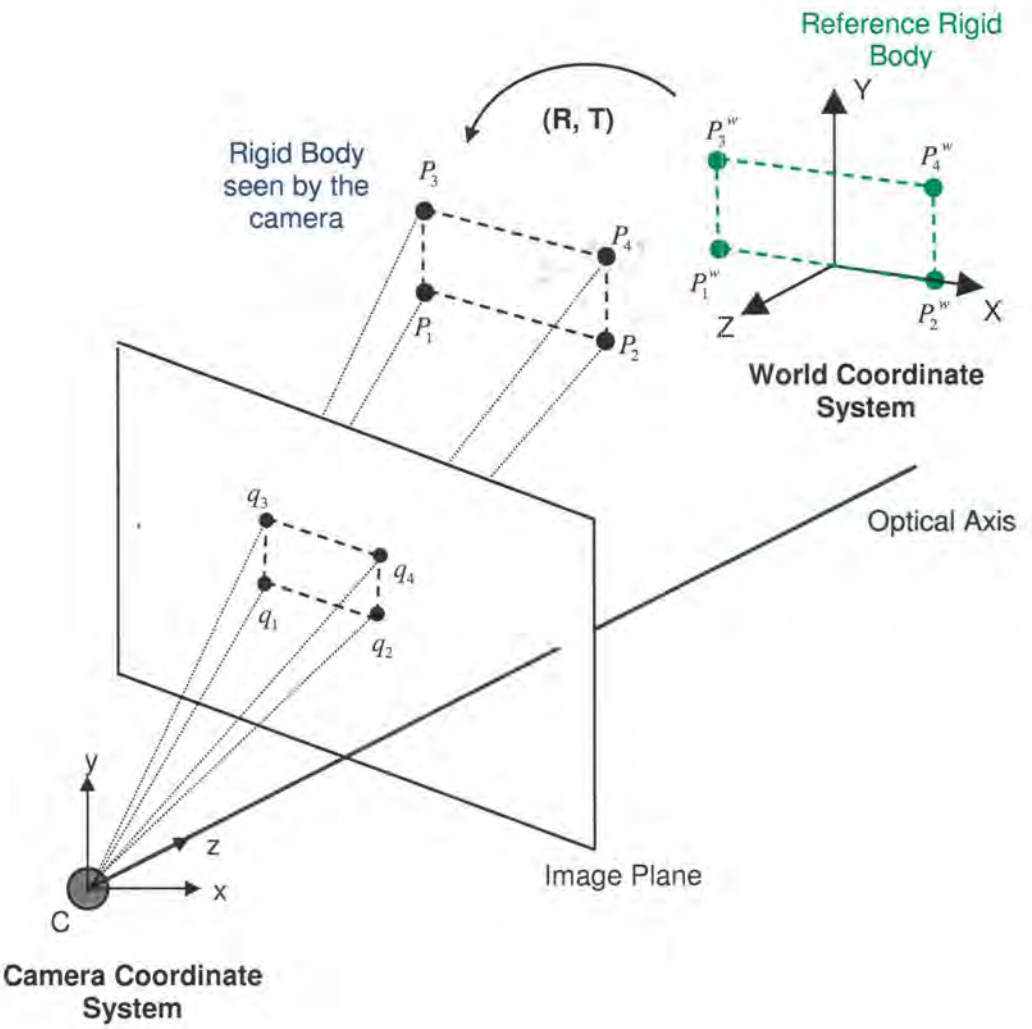


Figure 30 *Rigid Body Transformation between the reference rigid body and the one seen by the camera.*

However the available data to us are the image plane data (2D data). The relation from 3D camera coordinates (X_i, Y_i, Z_i) to ideal image coordinates (x_i, y_i) is established under pinhole camera model:

$$x_i = f\frac{X_i}{Z_i}$$
$$y_i = f\frac{Y_i}{Z_i} \qquad (41)$$

where f is the camera's focal length. If equations in (41) and (42) are combined we get these basic equations:

$$x_i = f(\frac{r_1 P_i^w + t_x}{r_3 P_i^w + t_z})$$
$$y_i = f(\frac{r_2 P_i^w + t_y}{r_3 P_i^w + t_z}) \quad (42)$$

These are the characteristic equations that associate the model rigid body data points with image plane data in terms of rotation and translation. The left side of these equations are the image plane data (x_i, y_i) that are delivered by the camera at each step and therefore they are known to us. The right side of the equations however, contain model rigid body coordinates P_i^w that is known a priori and used in the optimization step.

Therefore a transformation between two rigid bodies – the model rigid body and the one seen by the camera- leads to 6 unknowns, three of rotations (r_1, r_2, r_3) and three of translations (t_x, t_y, t_z).

By directly solving for these 6 unknowns, we get the absolute position and orientation of the rigid body that represents the Head or Camera Pose in the current frame without having to go through multiple steps as required in. The solution is obtained through an optimization problem which attempts to minimize:

$$\rightarrow \min \sum_{i=1}^{n} \| q_i - \phi(R, T, P_i^w) \|^2 \quad (43)$$

This is a nonlinear minimization problem, which can be solved with an iterative method like Gauss-Newton or the Levenberg-Marquardt algorithm. The iterative method will require an initial guess of the unknown rotation and translation parameters which can be obtained using a linear technique.

The minimum number of points required to solve the equation is n=3 points which makes the number of equations be the exact number unknowns:

$$n = 3 \rightarrow 3 \times (x_i, y_i) \quad known \rightarrow 6\, known$$
$$6 \quad unknown \quad (44)$$

If however n=4 points are used we get better accuracy of the results by the **over determined** system:

$$n = 4 \rightarrow 4 \times (x_i, y_i) \quad known \rightarrow 8\, known$$
$$6 \quad unknown \quad (45)$$

3 Implementation "Development of the Head Tracking Solution"

This chapter describes how the head tracking solution is implemented. Figure 31 illustrates how the head tracking modules are integrated into the 3D Audio Demo. The Wii remote is equipped with an IR camera that can capture the rigid body moving in the space. The rigid body has four mounted infrared LEDs that constitute an *IR-beacon*. In the case of 3D audio demo, the LEDs are mounted on the headphone worn by the listener and thus the head acts as a rigid body that should be tracked. The camera recognizes the IR-target points as bright spots in its image plane. The image plane is a two-dimensional (2D) plane. Each target point in the space has a three-dimensional (3D) coordinate and the 2D data recorded by the camera corresponds to the projection of the target points onto the camera's image plane according to the pinhole camera model. Hence, the position of the dots in the 3D space can be evaluated by inverting that projection.

The 2D image plane data is transmitted to the computer through a Bluetooth connection. The Wii remote driver library communicates to the Wii remote through Bluetooth stack. It delivers the raw tracking data to the head tracking library. The head tracking library receives the data and from that calculates the translation and orientation of the rigid body (user's head) in the space. How the head tracking library is obtained is explained throughout this chapter.

Subsequently, the head tracking results are delivered to the 3D audio demo. The 3D audio demo receives these data and produces binaural signals in full agreement with the head movement in the space. Therefore for an effective head tracked virtual audio environment, the integration of the head tracking solution with the 3D audio demo is required. How the above is achieved along with a tutorial on spatial audio is explained in next chapter.

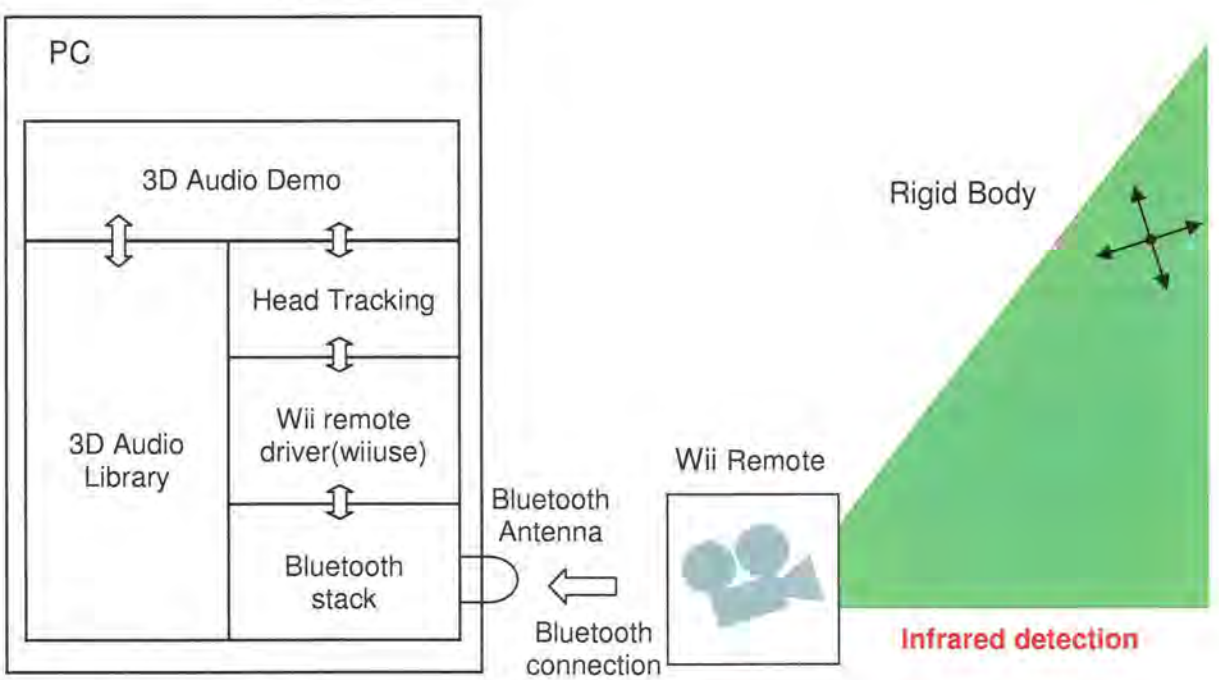


Figure 31 *The integration of the head tracking module into the 3D audio demo*

3.1 Setup requirements

According to figure 31, the idea behind this setup is to use the Wii remote controller as the tracking device. As already known, the Wii remote is equipped with an infrared tracker camera that is able to identify up to four infrared lights. The main idea now is to build a device with four infrared LEDs that can be recognized by the Wii remote camera. We call this device head tracking IR-Beacon.

3.1.1 Hardware requirements

1. The Wii remote controller

The Nintendo Wii remote will be used as the key tracking device. Nowadays, the controller can be purchased at a reasonable price of about 40 US dollars. The Wii sensor bar and the Wii console are not going to be used. We build or own custom IR-beacon in place of the sensor bar and the communication is established directly with the computer.

2. Bluetooth dongle

A Bluetooth dongle is needed to realize the physical Bluetooth connection. Some Bluetooth dongles ship their own Bluetooth stacks

3. A rigid body with four IR-LEDs

In order to track the movements of a rigid body in six degrees of freedom (6DOF), the rigid body is fitted with four infrared light emitting diodes (IR-LEDs). The positions of those diodes are then tracked by the Wii remote.

A solderless breadboard is used as a rigid body.
Using a breadboard, it is easy to change the configuration on any arrangement without doing any soldering. In the end, the LEDs will be mounted on wireless headphone as one of the components of a virtual audio environment.

As discussed in section Band pass filter and Blob detection, the Wii camera has the maximum radiation power reception at wavelength about $\lambda_{Max-IR} = 940nm$. Therefore, LEDs are chosen to have wavelength at peak emission nearly closed to this wavelength:

$$\lambda_{Max-IR} \approx \lambda_{peak-Wii} = 940nm \quad (46)$$

While choosing the IR-LEDs it is important to look out for LEDs with a wide range of angle of radiation. We chose the LED model (OP165W IR-LED 0.5mW 940nm 3.1mm) because it has a very wide Beam Angle (±45 °), compared to typical IR-LEDs (figure 32)

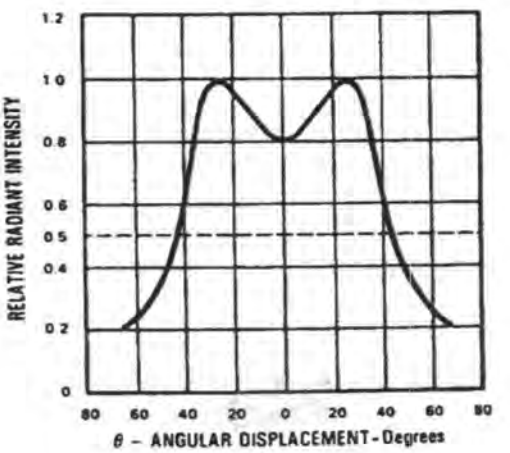


Figure 32 *Relative radiant intensity vs. Angular displacement*

The configuration uses a 1.2 Volt AA battery block as the power source, a standard SPST switch and wires to connect the LEDs. To save the energy no resistor is used. The LEDs are connected in parallel according to the figure below:

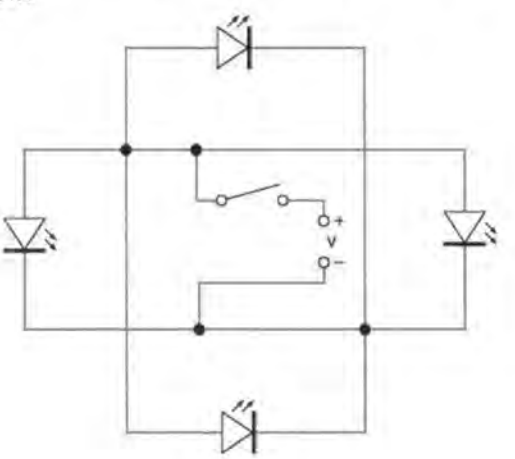


Figure 33 *The diagram for the IR tracking beacon circuitry*

It is very important that the four target points are NOT all in the same plane. This is a requirement for the head tracking algorithm to converge. Three of four LEDs are therefore aligned to have nearly similar height but the forth LED is mounted higher than the others as shown below:

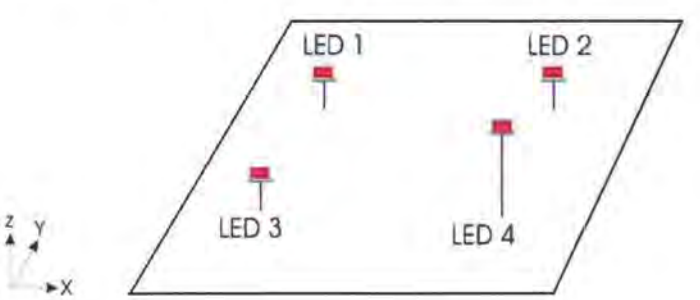


Figure 34 *Schematic of an IR-LED beacon. Note that the fourth LED has a different height level compared to the others.*

Once the LEDs are mounted on the beacon, the exact distance between each two LEDs should be measured and saved. These measurements are used as the reference data and are required to evaluate the 3D position of IR dots. It is very important to measure these quantities very accurately otherwise the tracking results will be poor. The values are measured in millimetres.

The IR-Beacon used in our work has the following measurements:

```
d12=127;        % [MM]
d13=27;         % [MM]
d24=33;         % [MM]
d34=129;        % [MM]
d14=132;        % [MM]
d23=127;        % [MM]

d=[d12,d23,d34,d14,d13,d24] ;  % [MM]
```

As could be seen, the main components of an IR-Beacon suitable for 6DOF tracking are four IR LEDs that constitute a non-coplanar arrangement and a power source. A picture of our current IR-Beacon is shown in figure 35. In the final solution, the diodes will be mounted on a wireless headphone as one of the components of Virtual Audio Environment.

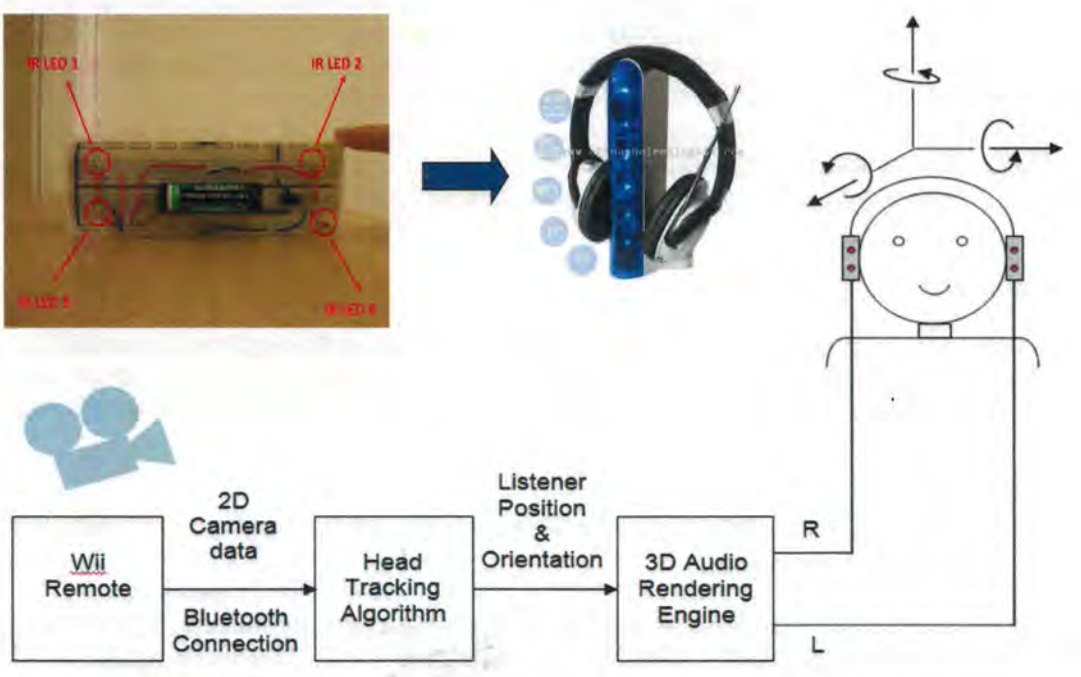


Figure 35 *Photograph of our IR-Beacon. In the final solution, the diodes will be mounted on a wireless headphone as a component of virtual audio environment.*

3.1.2 Software requirements

1. The Wii remote driver library

An effort has already been made to develop a library for communicating with the Wii remote. With a library available, we can focus our study on the head tracking algorithm instead of how to interface with the Wii remote. Wiiuse is a library written in C and chosen for this project since our head tracking software is also complied as a library in C.

The wiiuse library offers a clean and light API that is single threaded and non-blocking. Wiiuse can connect to several Wii remotes and is chosen in this project. It supports motion sensing, IR tracking and other features. Wiiuse is an open source library that supports Windows and Linux.

2. The Bluetooth stack

A Bluetooth stack is required to establish a Bluetooth connection. A variety of Bluetooth stacks are available on the market that can do the job. For Linux, there is well known stack called BlueZ which has an easy implementation. For windows, there is a variety of different Bluetooth stacks. There are native Microsoft stacks, Bluetooth stacks and Widcomm stack. Windows XP service pack2 has already the Bluetooth stack implementation.

3. MATLAB

Simulation plays a key role in this project and MATLAB is the main simulation software used for verifying the performance of the algorithm. MATLAB is a high-level language and has an interactive environment that enables us to perform computationally intensive tasks faster than with traditional programming languages such as C and C++. We use MATLAB for two purposes:

1. Evaluation of the algorithm based on simulation
2. Evaluation of the algorithm based on recorded data

4. Visual studio

The final head tracking solution is developed as a library in C. Visual Studio 2008 is thus chosen to compile as the integrated development environment (IDE). Further explanation about library realization is explained can be found in section "Realizing the library".

3.2 Design

3.2.1 Problem formulation

The Wii remote exploits an infrared camera to support the infrared tracking feature. The Wii remote tracker is as an optical marker-based tracker. It is optical because the Wii contains a camera and it is marker-based because the camera does not report the whole image that camera sees. Instead, the Wii remote delivers the position of up to four infrared LEDs which represent our markers. The camera delivers the position of four LEDs in 2D coordinates in pixels which represent the markers detected by the camera (see figure 36). Technically, these coordinates represent the projection of the LEDs onto the camera's image plane according to the camera projection equations (see The pinhole camera model). The projected data are sometimes called *image plane data*.

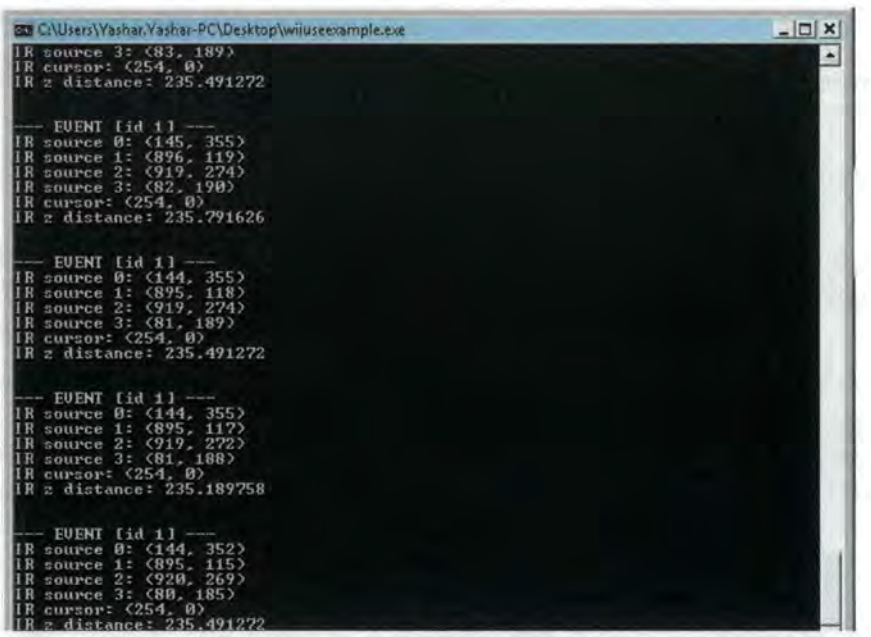


Figure 36: *Snapshot of screen during tracking. Position of IR sources (0-3) (markers) is delivered by the camera at each instance of tracking. These coordinates are 2D coordinates and represent the projection of the IR-LED onto the camera's image plane in units of pixel where* $x \in (0,1023)$ *and* $y \in (0,767)$

The main task is now to use the image data and the known positions of the LEDs of the reference IR-beacon to calculate the position and orientation of the IR-LED beacon in 3D space. This task is known as Pose estimation and is a common problem in computer vision. The pose estimation problem for this project can be formulated as below:

- GIVEN:

(x_i,y_i) Position of the target points in the camera image plane

(x_i,y_i,z_i) Positions of the LEDs of the reference IR-beacon

- FIND:

Translation[11] and Orientation of the rigid body in 3D space in 6DOF

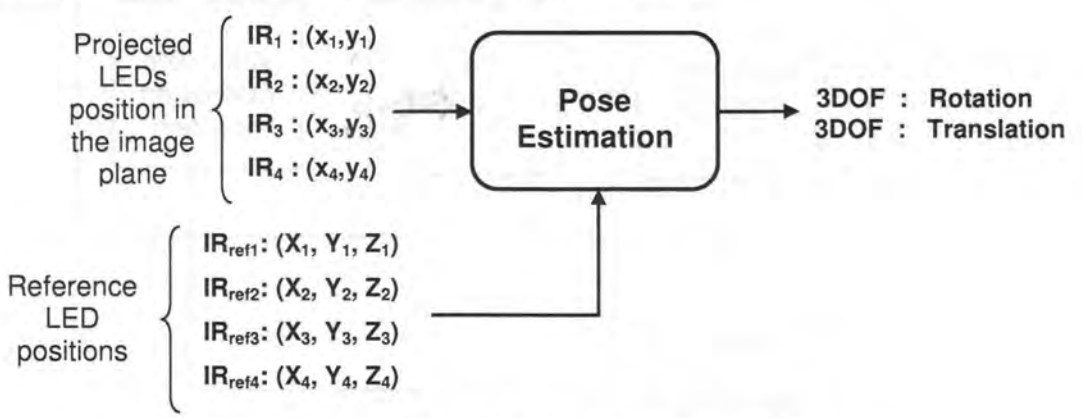


Figure 37: *Problem formulation for pose estimation*

[11] Note that Translation is directly related to Position.

3.2.2 Constraints

- The fact that the Wii remote camera can track only up to *four* points limits the choice of methods. If more points could be detected by the camera, more accurate methods with robust performance could be investigated.
- The quantization noise of the camera can affect the performance results and is key source for inaccuracies of the measurements.
- The field of vision of the camera as well as LEDs limited beam angle restricts freedom of IR-Beacon in its movement in 3D space.

3.2.3 System model

Back to our discussion earlier on Pose estimation methodologies, the central task is to find a method to estimate the pose (position and orientation) of the rigid body from 2D image plane data by knowing configuration of the rigid body a priory.

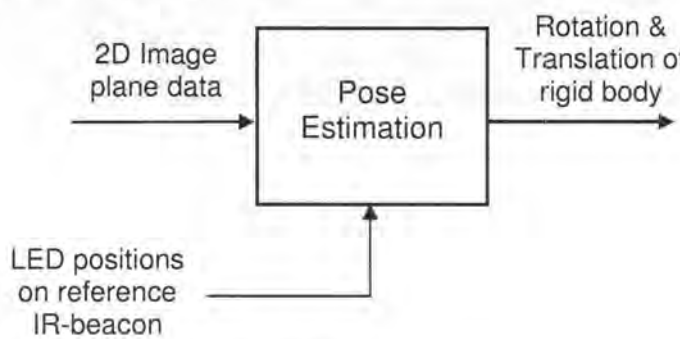


Figure 38 *Pose Estimation*

Some of the possible algorithms for this purpose are suggested on section Pose estimation methodologies. We use the 2-step method to solve for the unknowns that is first the distances between the camera and each target point is calculated. The results are then used to reconstruct the LED positions. By comparing the acquired rigid body and a known reference rigid body, position and orientation is calculated. Figure below illustrates this process:

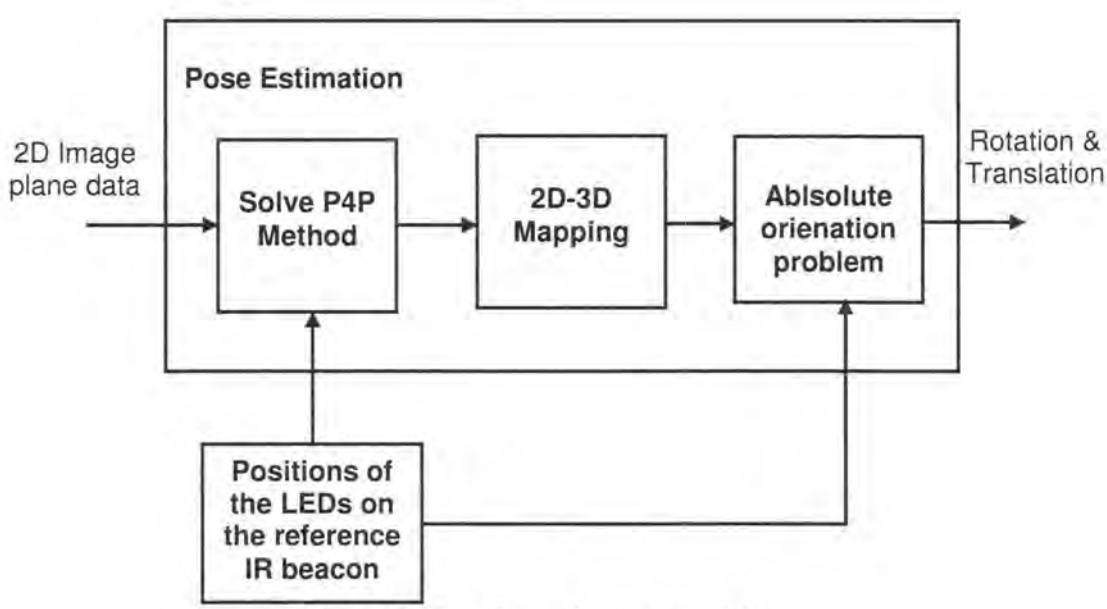


Figure 39 *Pose Estimation main modules*

The complete information flow in our head tracking algorithm is presented below. In the following sections, a detailed description of each module in our head tracking algorithm will be given:

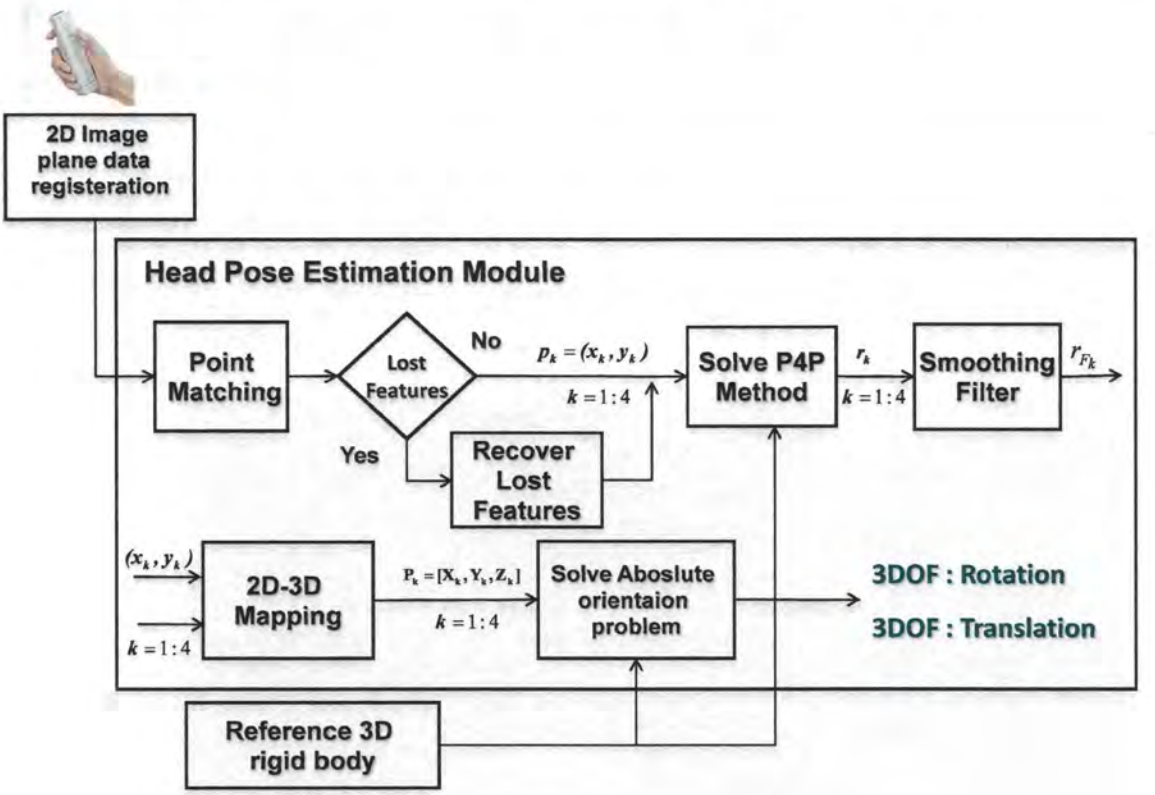


Figure 40 *The information flow in our head tracking Algorithm*

3.2.4 Reference 3D rigid body

Prior to tracking a set of 3D model points is acquired and saved. The model set serves as the reference data during the tracking process. When the 3D data are recovered, they are need both in the PnP method and the rotation and translation evaluations. Figure 41 illustrates the model for our dot arrangement in which the world coordinate system is assumed to be centred right at middle point of the dots and the reference points are measured from this origin:

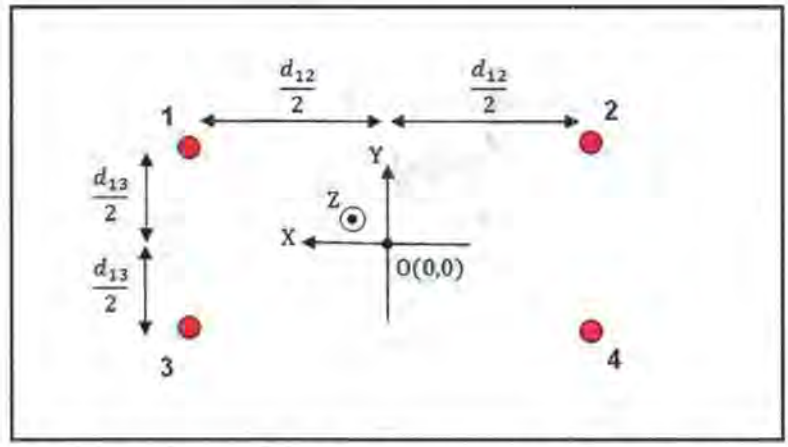


Figure 41 *The reference points in the world coordinate system (top view).*

The reference points for the above arrangement are as follow:

```
pref1=[d12/2,d13/2,3]';          % [MM]
pref2=[-d12/2,d24/2,3]';      % [MM]
pref3=[d34/2,-d14/2,3]';         % [MM]
pref4=[-d34/2,-d24/2,24]';       % [MM]

PREF=[pref1,pref2,pref3,pref4];   % [MM]
```

The last entity of the above coordinates (3, 3, 3, 24) shows the height of the dots in z direction.

3.2.5 2D image plane registration

During the tracking, the camera views the moving target point and recognizes them as bright spots in its 2D image plane. The features are the projected points from 3D space to 2D space using the camera projection equations (see The pinhole camera model). After that the Bluetooth connection is established and the controller and the computer are paired, the ⌂ button on the Wii remote should be pressed once to start the IR-tracking function.

Once the tracking is started, the following information about each IR LED is received.

LED Number	X Coordinate	Y Coordinate
<0-3>	<0-1023>	<0-768>

Figure 42 *Data reported by the camera*

The LED enumeration is done arbitrarily by the wiiuse but once the LEDs are identified, the software will keep the same LED enumeration for as long as the four diodes remain within the camera's field of vision. Nevertheless, we need to modify the enumeration of the LEDs based on our own configuration as shown in figure 41. This is because the priori saved distances d_{ij}s have to be matched with the right distances so that the calculations are correct. The procedures for matching the diodes are explained in the subsequent section.

The Wii remote supplies the updates of information for as many as 100 times per second. It is very important that during the actual tracking, all four bright spots are within the camera's field of vision and all the target points are detectable by the camera at each tracking update. This is because our head tracking solution is based on the P4P method which requires four pairs of (x,y) coordinates to be delivered to the head tracking algorithm. IF for any reason some of the features are not detected at some updates, the whole group of data for that update is ignored until a complete package is received. Nevertheless, these enumerations are useful to separate each group of the data from the others, at different updates.

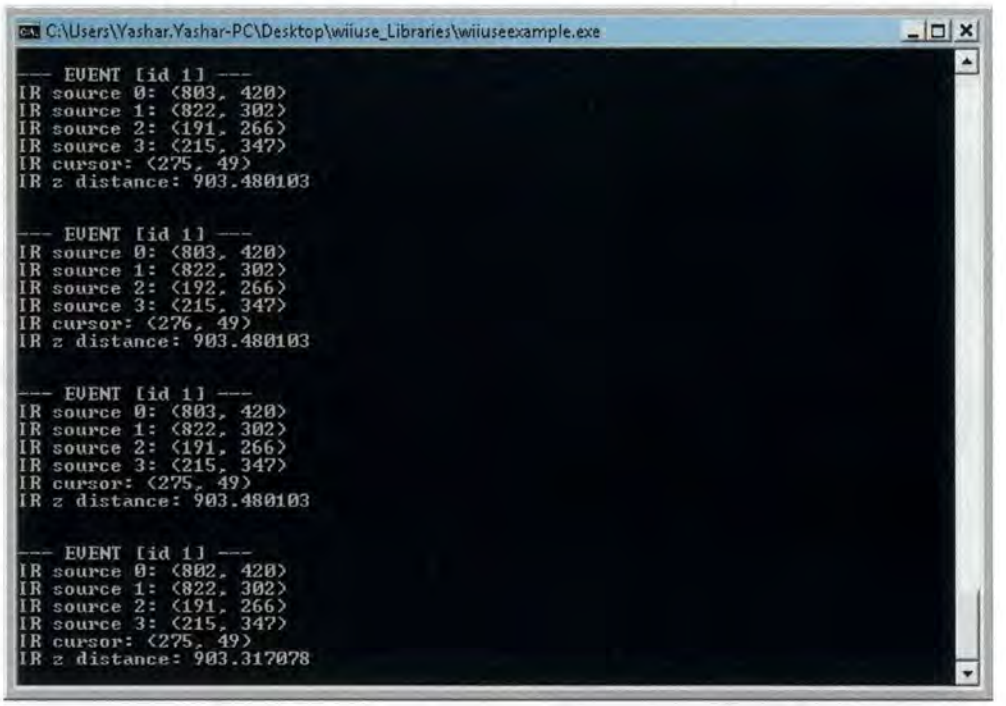


Figure 43 *Wii remote in tracking mode. The camera keeps the enumeration the same during the tracking as long as the bright spots remain in the camera's field of view.*

3.2.6 Point matching

The camera only reports up to four bright spots it recognizes on its image plane, with no particular order and no association to the LEDs on the IR beacon. The point matching thus refers to the idea of associating the target points in 3D space and their projections in the image plane of the camera.

For example in figure 44, based on the coordinates received, the camera is reporting the configuration on the left hand side whereas the configuration should be rearranged to have the enumeration on the right picture for all the updates:

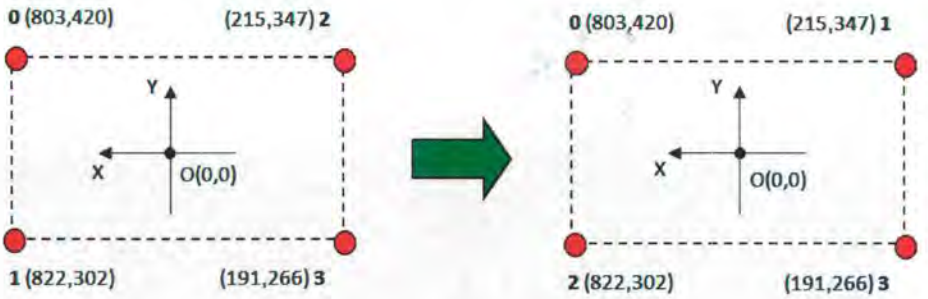


Figure 44 *Association of the data reported by the camera with the LEDs on our IR beacon. The arrangement reported by the camera (Left), the enumeration of the data for use in our algorithm (Right)*

If the camera is at rest that is relatively horizontal and orthogonal to the IR beacon, this arrangement could be done fairly easily. This is a simple clustering problem in two dimensions that could be done in two steps as depicted in the figure below and described in the next page:

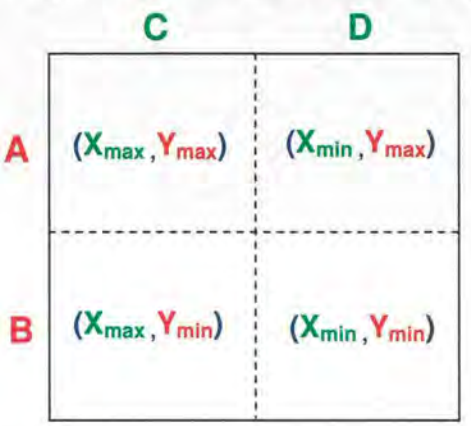


Figure 45 *Data clustering of the received data*

1. Divide the data into two groups based on the y coordinates. So for example in the above case we have:

Group A: (803,420) , (215,347)

Group B: (822,302) , (191,266)

2. Divide each group into two subgroups based on their x coordinates:

Group A:

- Group C : (803,420) , Group D : (215,347)

Group B:

- Group C : (822,302) , Group D : (191,266)

The above procedure is done once at the beginning of the tracking and when the IR beacon starts rotating and changing its posture, then this idea cannot be used to identify the LEDs.

One alternative could be to take benefit of the fact that the updates are reported every 10 ms. In such a small fraction of time, two consecutive groups of data cannot have significant difference in their values, that is to say the same dots remain close to each other. Thus, we can track the spots by evaluating the distance between point i in the update k and all the other points in update k+1 and choosing the smallest distance as the corresponding point in group k+1. This should be done for all the other points in group k.

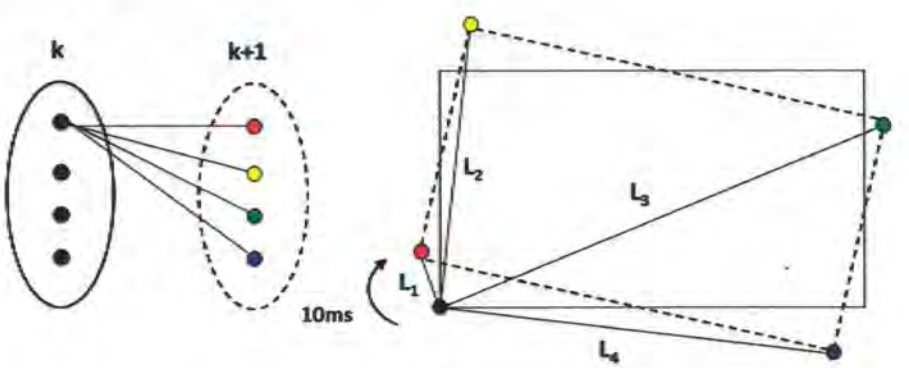


Figure 46 *Point matching at each update. Choose the smallest distance* $L_1<L_2<L_3<L_4$

3.2.7 Feature recovery

It might happen that during the tracking, some of the features are not reported by the camera. This is the case when the camera cannot detect some features when they remain outside camera's horizontal or vertical field of vision. In this case, the head tracking algorithm cannot evaluate the head pose because at least four features are required for the algorithm to perform.

However, taking into account that the configuration model is known a priori, if and only if one feature is not detected by the camera, the problem could be alleviated by using a prediction model. The prediction model estimates the fourth feature based on the previous information so that we can predict the missing point and use it estimate the pose.

The linear prediction models however, usually predict the values of discrete-time signal based on previous samples. Given the fact, if the fourth feature is not visible for a significant amount of samples which is the case when the camera is updating information rapidly, the predictions for the fourth spot place should be made based on previously made predicted features which apparently is not a precise prediction.

A better idea is to use the PnP method for given n=3 known. We know that P3P is a reduced version of P4P method and is solvable under certain considerations. Thus once P3P is solved, the three features in 3D space are recovered, by knowing the model of our configuration, the fourth point is predicted:

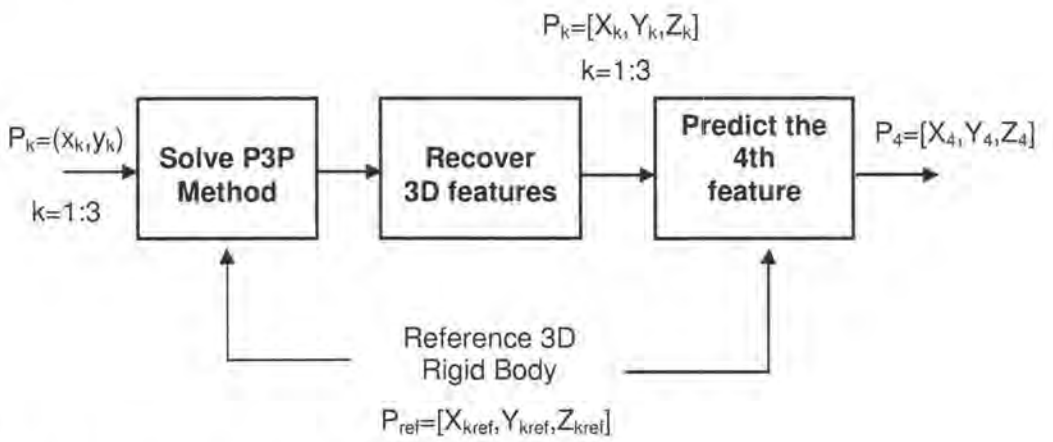


Figure 47 *The modules required for feature recovery*

If more than one feature is missing, it is not possible to recover them using this method. The feature recovery module is currently switched off in our implementation.

3.2.8 Solve P4P method

1. Choosing the equation system

The Wii camera reports four pairs of coordinates at each tracking update (every 20ms) for which an estimation of head pose in three-dimensional space has to be made. The common methods used for this purpose are The perspective N-Point Problem (PnP) and Direct Rigid Body Transformation discussed in section "pose estimation methodologies".

The PnP method aims at finding camera-point distances, the distance from the camera's centre to each bright spot. Once the 3D coordinates are available, the position and orientation of the rigid body can be evaluated. The Rigid Body Transformation however, seeks to estimate the pose directly by

formulating the equations between the position and the orientation of two rigid bodies. Solving for those unknowns directly gives us Rotation and Translation without having to go through multiple steps.

Accordingly, in this project a fair amount of study was done on the selection of method that is well suited for our problem. We ended up choosing the perspective N-Point method because it has a straight forward problem formulation and the types of nonlinearities that exist in the equations seems to be solvable fairly accurately and rapidly with standard iterative methods.

The pose estimating method (the PnP method) was first simulated in MATLAB and its sensitivity was analyzed with respect to other parameters. A random walk in 3D space for the IR-Beacon was performed to measure the accuracy of the results. Afterwards, when reasonable results were obtained in terms of accuracy, convergence and speed, it was chosen for implementation in C for real-time processing.

The reminder of this section discusses the equation system and optimization algorithms chosen:

Four unknown points $P_1=(X_1,Y_1,Z_1)$, $P_2=(X_2,Y_2,Z_2)$, $P_3=(X_3,Y_3,Z_3)$, $P_4=(X_4,Y_4,Z_4)$ in the space are seen by the camera at the origin. The projection of these points onto the camera's image plane are captured by the camera. We denote these projections $q_1=(x_1,y_1,f)$, $q_2=(x_2,y_2,f)$, $q_3=(x_3,y_3,f)$, $q_4=(x_4,y_4,f)$ and f is the camera's focal length, the distance from the origin to the image plane.

The cosine of angles between the rays to the points can also be found:

$$\cos\theta_{ij} = \frac{< q_i, q_j >}{|q_i||q_j|} \tag{47}$$

The distances d_{12}, d_{13}, d_{14}, d_{23}, d_{24}, d_{34} are also known beforehand.

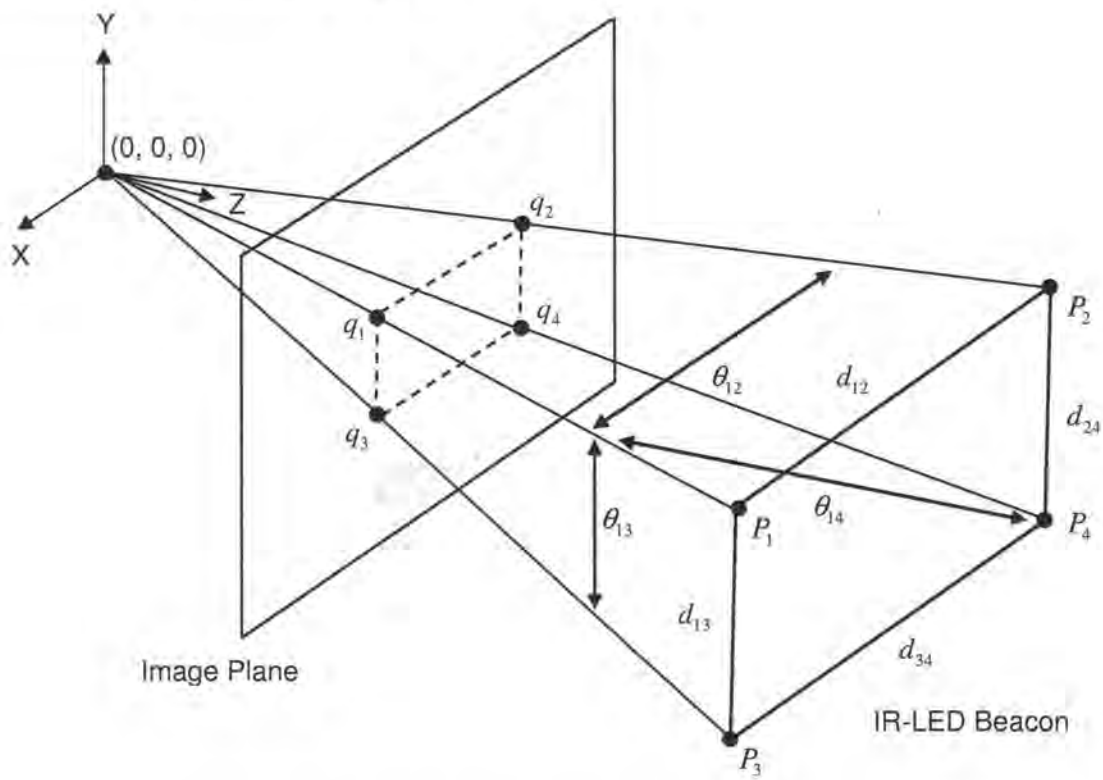


Figure 48 *Solving Perspective 4-Point Problem for our configuration*

Now using the law of cosines, the distances r_1, r_2, r_3, r_4 from the origin can be demined be solving the system of quadratic equations between each two pairs of points in the 3D space where r_i is the distance from *i*'s point to the origin.

$$\begin{aligned} r_1^2 + r_2^2 - r_1 r_2 \cos\theta_{12} &= d_{12}^2 \\ r_1^2 + r_3^2 - r_1 r_3 \cos\theta_{13} &= d_{13}^2 \\ r_1^2 + r_4^2 - r_1 r_4 \cos\theta_{14} &= d_{14}^2 \\ r_2^2 + r_3^2 - r_2 r_3 \cos\theta_{23} &= d_{23}^2 \\ r_2^2 + r_4^2 - r_2 r_4 \cos\theta_{24} &= d_{24}^2 \\ r_3^2 + r_4^2 - r_3 r_4 \cos\theta_{34} &= d_{34}^2 \end{aligned} \tag{48}$$

The function **RadiusRcd** calculates the cosine of angles between the rays to the points, for given image point data q_i. During simulation q_i is generated randomly.

Figure 49 *The RadiusRcd function computes the cosine of the angles considering noise effect on the input image plane data*

In addition, as explained in section "Intrinsic camera parameters", the real resolution of the camera seems to be $128 * 96$ and the camera uses 8 interpolation levels to reach the resolution of $1024 * 768$. This effect can be modeled by a quantization noise that is imposed by the camera to the equations system.

Therefore, the effect of the quantization in the final results should be analyzed and thus the simulation process is performed to investigate how the quantization noise propagates through the equation systems and how accurate are the results. The function **RadiusRcd** can be then used to calculate the ray angles for the noisy image plane data. It maps raw captured data to the nearest integer of multiple eight according to a uniform quantization (see "Simulation of the camera capture"). The results are used in equation in PnP methods evaluation.

2. Choosing optimization algorithm

The first step to evaluate the 6DOF pose is to evaluate the system of over-determined equations as described in Eq. 4.3. The full system of six nonlinear equations can be solved using a variety of methods such as *Levenberg-Marquardt* or *Gauss-Newton*. Gauss-Newton method is a minimization method that works very well in practice and is selected as the optimization method (see Gauss-Newton algorithm).

The basic idea is, given distances d_i between the dots and calculated cosine of angles c_i, to minimize the residual distances in two sides of Eq 48. A starting value r_0 for the unknowns is required to initialize the algorithm.

In order to improve the efficiency of the algorithm in terms of speed and robustifiy its convergence, the iteration converged results are fed back into the algorithm as the next initial values as depicted in the figure above.

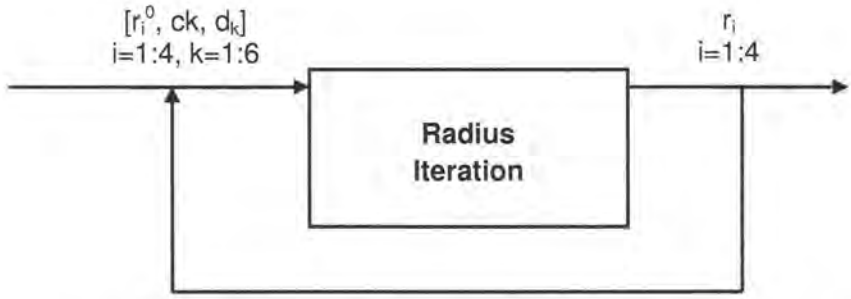


Figure 50 *The Radius Iteration function computes the camera-point distances*

According to Gauss-Newton method, the Jacobean matrix should be computed at each iteration.For the system of Equation in 48, the Jacobian matrix is a 6×4 matrix.

Subsequently, a least square problem should be solved. In MATLAB the backslash (\) operation finds the least-square solution to the system of over-determined (or under-determined) equation using QR decomposition. The operation automatically handles the rank deficiency problems. In C however, a program should be written to solve the least square problem. The classical *Schur* algorithm is used to obtain the solution. The iterative algorithm stops when the algorithm converges to the optimum point before the maximum number of iteration is reached. See figure 51:

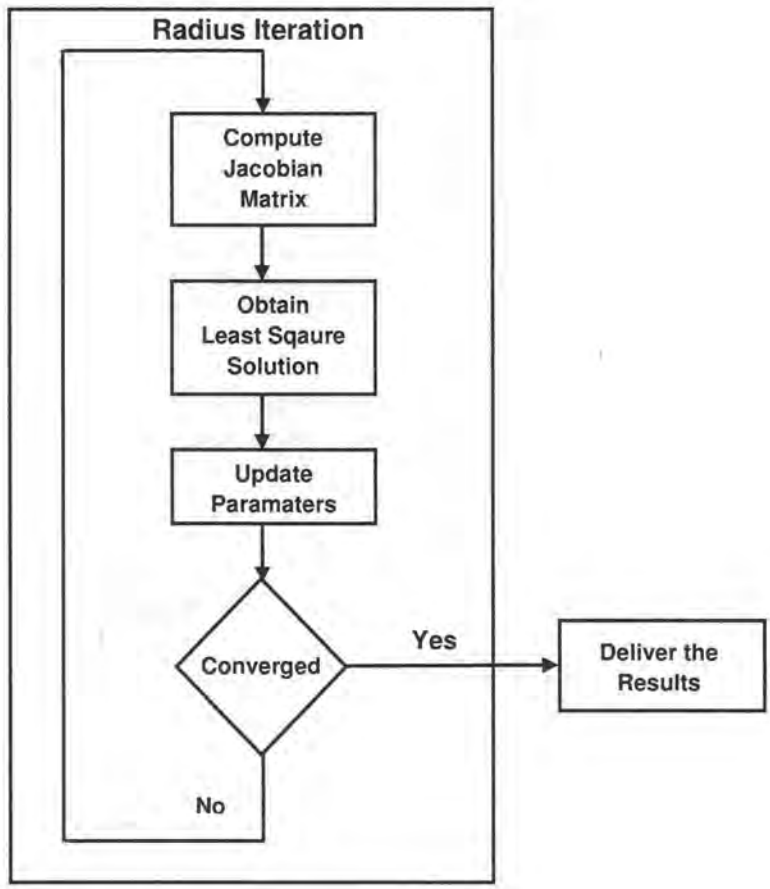


Figure 51 *The Radius iteration procedure*

3.2.9 Smoothing filter

A smoothing filter is implemented to remove noise from the computed data. It is placed right after P4P module so as to filter out the data before errors gets propagated into the following modules. But why a smoothing filter is necessary to be used in the head tracking algorithm?

The filter is meant to detect possible outliers during the computations due to divergence and remove them from the data. The iterative algorithm uses the last iteration result as the next starting point and if a divergence case happens and no filter is used, that false result will be passed to the next iteration that might cause it to diverge as well. Accordingly, the errors might propagate to the next runs and direct them into divergence. So filtering is not only important for the current data block but also it somehow avoids errors in the result.

The best filter for detecting the outliers (high frequency components) is a **median** filter which as its name implies, replaces the value of a sample by the median of values in the neighbourhood of that sample. Median filter is a nonlinear[12] filter used to perform high degree of noise reduction. The median filter is suitable because it is not sensitive to the level of noise data.

A median filter of size five is used as the smoothing filter. The data are buffered and shifted through the median filter

Figure 52 *A median filter of size 5 used as smoothing filter*

3.2.10 2D-3D mapping

The next step after computing the camera-point distances is to calculate the three-dimensional coordinates of the target points. The goal is to match the 2D feature points $q_i = (x_i, y_i)$ with the corresponding 3D target $P_i = (X_i, Y_i, Z_i)$. The geometric illustration of this problem is depicted in figure 53 where *f* is the camera's focal length.

V_i is the vector representing the dot directions originating from the camera centre. The 2D feature points lie in the direction of this vector and thus a relationship between 2D and 3D correspondence can be derived according to the formulas presented below:

$$
\begin{aligned}
V_i &= \frac{x_i\vec{i} + y_i\vec{j} + z_i\vec{k}}{\sqrt{x_i^2 + y_i^2 + z_i^2}} |r_i| \\
X_i(mm) &= \frac{x_i(pixel)}{\sqrt{x_i^2 + y_i^2 + z_i^2}} |r_i(mm)| \\
Y_i(mm) &= \frac{y_i(pixel)}{\sqrt{x_i^2 + y_i^2 + z_i^2}} |r_i(mm)| \\
Z_i(mm) &= \frac{z_i(pixel)}{\sqrt{x_i^2 + y_i^2 + z_i^2}} |r_i(mm)|
\end{aligned}
\qquad (49)
$$

[12] Known as an order-statistic filter type because its response is based on ordering (ranking).

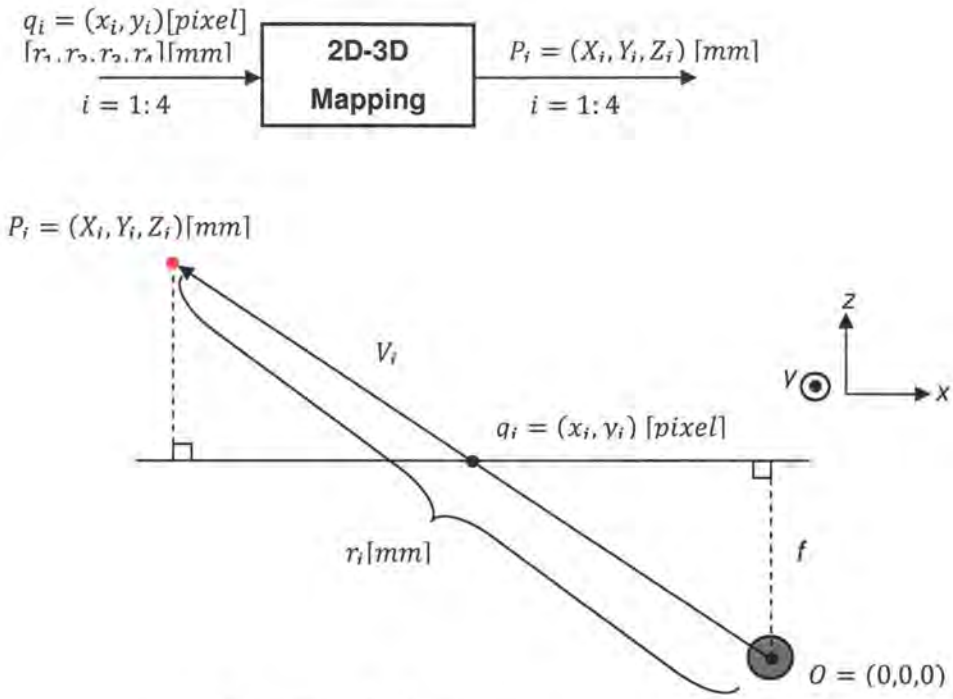


Figure 53 *2D-3D mapping*

3.2.11 Solve absolute orientation problem

Up until know, we have been able to reconstruct the moving rigid body. So 3D coordinates of the moving rigid body $P_i = [X_i, Y_i, Z_i]^T$ acquired. On the other hand, a reference 3D rigid body $P_i^{ref} = [X_i^{ref}, Y_i^{ref}, Z_i^{ref}]^T$ is known and saved from the beginning. Now the task is by comparing these two sets of data obtain the rotation and translation which rotates R and translates T the reference model $P_i^{ref} = [X_i^{ref}, Y_i^{ref}, Z_i^{ref}]^T$ to match the recovered model $P_i = [X_i, Y_i, Z_i]^T$.

$$P = RP^{ref} + T \tag{50}$$

The problem of finding R and T is a well known problem in computer vision and is known as the *absolute orientation problem*. The problem is formulated as a least-square problem as below:

Given to point sets P_i, P_i^{ref} for $i = 1:N$ related by:

$$P_i = \underbrace{RP_i^{ref} + T}_{P_i^{tr}} + N_i \tag{51}$$

where N_i the noise vector, find the best R and T which minimizes the following least square formulation:

$$\Sigma^2 = \sum_{i=1}^{N} \left\| P_i - (RP_i^{ref} + T) \right\| \tag{52}$$

The solution can be acquired using an iterative or non-iterative algorithm. We use a non-iterative algorithm based on singular value decomposition (SVD) to solve the least-square problem. The solution is attained by decoupling Rotation and Translation and solving for them individually in two steps. See [15] for further information.

3.2.12 Decoupling translation and rotation:

It is shown in [13] that if $\hat{R}$ and $\hat{T}$ is the solution to 52 then $\overline{P} = \overline{P}^{tr}$ where:

$$\overline{P} = \frac{1}{N}\sum_{i=1}^{N} P_i$$

$$\overline{P}^{ref} = \frac{1}{N}\sum_{i=1}^{N} P_i^{ref} \tag{53}$$

$$\overline{P}^{tr} = \frac{1}{N}\sum_{i=1}^{N} P_i^{tr} = \frac{1}{N}\sum_{i=1}^{N}(RP_i^{ref} + T) = \hat{R}\overline{P}^{ref} + \hat{T}$$

Assume

$$Q_i = P_i - \overline{P} \Rightarrow P_i = Q_i + \overline{P}$$
$$Q_i^{ref} = P_i^{ref} - \overline{P}^{ref} \Rightarrow P_i^{ref} = Q_i^{ref} + \overline{P}^{ref} \tag{54}$$

Eq. 52 is then equal to:

$$\Psi^2 = \sum_{i=1}^{N}\left\|P_i - RP_i^{ref} + T\right\| = \sum_{i=1}^{N}\left\|(Q_i + \overline{P}) - (\hat{R}(Q_i^{ref} + \overline{P}^{ref}) + \hat{T})\right\|$$
$$= \sum_{i=1}^{N}\left\|(Q_i + (\hat{R}\overline{P}^{ref} + \hat{T})) - (\hat{R}(Q_i^{ref} + \overline{P}^{ref}) + \hat{T})\right\| \tag{55}$$
$$= \sum_{i=1}^{N}\left\|(Q_i - \hat{R}Q_i^{ref})\right\|$$

The original least square problem is minimized to finding $\hat{R}$. Translation $\hat{T}$ accordingly when $\hat{R}$ is know.

1. Find the Rotation $\hat{R}$ that minimizes Ψ^2 in Eq. 55
2. Find the Translation by:

$$\hat{T} = \overline{P}^{tr} - \hat{R}\overline{P}^{ref} \tag{56}$$

4 3D Audio Integration

In the previous chapter, it was described how the head tracking algorithm is created. The output of the algorithm is the coordinates of the head in the 3D space. The raw coordinates might not be so interesting, so in order to be effective, the head tracking applications are implemented and integrated with the head tracking library to show the applicability of the program in practice. The main application of the head tracking application is in virtual reality and 3D audio rendering engine is an audio virtual environment which can be integrated with the head tracking library to render head tracked-based 3D audio scenes. In such a scenario the user's head will act as an input device to the application.

This chapter is divided into two main sections. The first section gives a tutorial on *spatial hearing* and describes the 3D audio and acoustic environment modeling technology developed by Ericsson Research which is the result of extensive research and development in the audio field.

The second part provides a general overview of how the head tracking library is realized and integrated with the 3D audio rendering engine.

4.1.1 Overview of spatial hearing

The world we live in is three dimensional and our ability to make sense of this spatial environment, operate within it and interact with it depends strongly on our spatial awareness, which we obtain through our spatial perception derived from our vision and our hearing.

We rely more on our vision to perceive the spatial scene in front of us, but for things that are not directly in front of us we rely more on our hearing.

After a long evolution we humans are now equipped with a highly developed spatial hearing that is able to give us a fairly detailed picture of our spatial environment deduced only from the acoustical signals reaching our two ears.

One of the most important components of our spatial hearing is the one that enables us to sense from which direction a sound wave that is hitting us is coming from and that we are able to separate several such simultaneous sound waves.

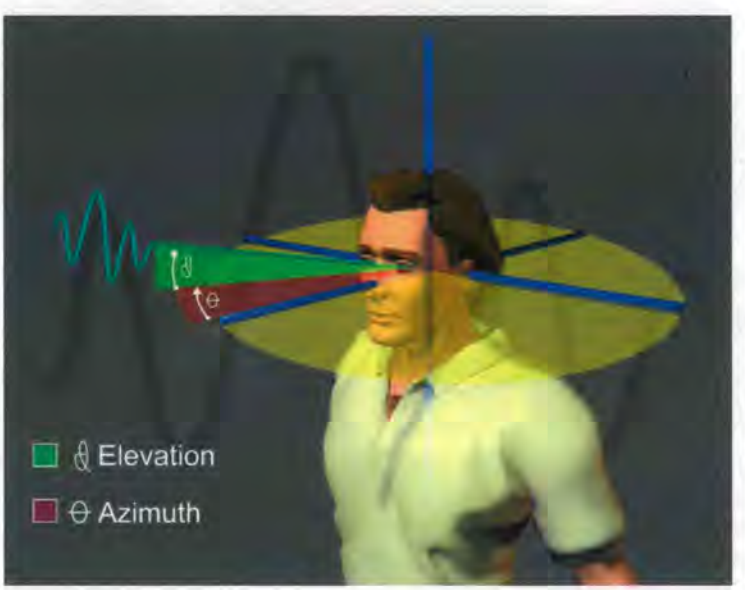


Figure 54 *Through our spatial hearing we are able to sense from which direction a sound wave that is hitting us is coming from.*

This ability to separate several sound sources approaching us from different directions is much better than our ability to separate them if they were all coming from the same direction. In a cocktail party situation we are able to use this ability to focus our attention on a single conversation of all the ones that are taking place around us.

In a room or some other enclosed spatial environment a sound wave that is emitted into the environment will be reflected several times by the walls, floor, ceiling and other reflective surfaces in the environment before dying out. When we are in such an environment we will receive several filtered and delayed versions of the same sound. The accumulated signal of these filtered and delayed sounds is usually referred to as the reverberation of the original sound. From this reverberation signal our spatial hearing is able to give us a sense of the approximate size of the room or enclosure we are in (i.e. a small room versus a concert hall) as well as the reflective nature of the surfaces in the environment (i.e. small office versus a tiled bathroom). These are also important pieces of the picture of our spatial environment that we obtain from our spatial hearing.

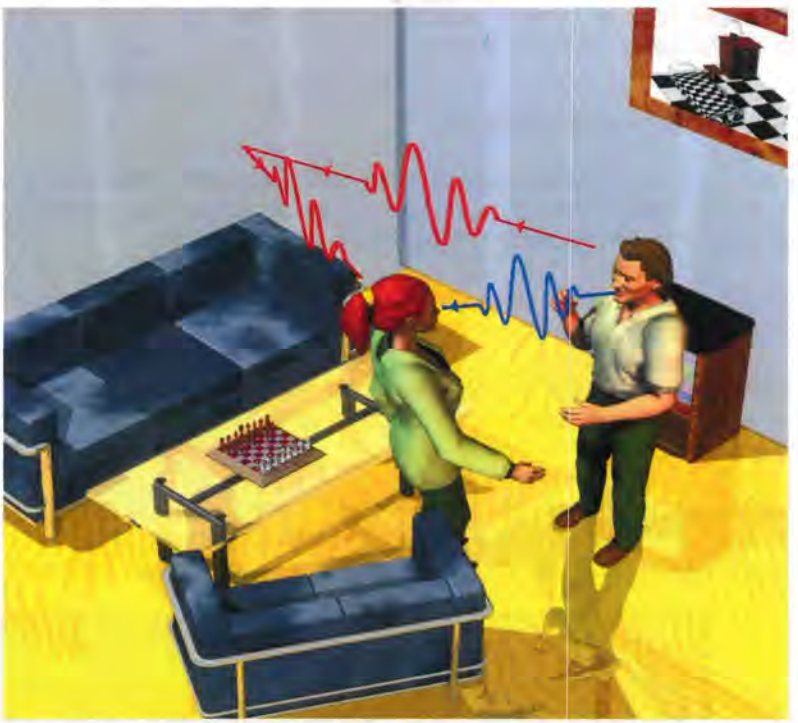


Figure 55 *In a room a sound wave that is emitted into the environment will be reflected several times by the walls, floor, ceiling and other reflective surfaces in the environment before dying out.*

4.1.2 Virtual 3D audio environments

The only signals available to our hearing are the signals picked up by our two ear drums and all the spatial information provided by our spatial hearing are deduced from those two signals.

Creating a virtual 3D audio environment for a person is therefore about simulating the left and right ear drum signals that person would hear in the virtual environment and deliver those simulated signals to that person's left and right ear. That person is usually referred to as the observer or listener of the environment.

This simulation is based on having a geometric description of the virtual spatial environment, information about the movements of the sound emitting sources, the listener and other "living creatures" and objects within the environment and information about the reverberation inside the environment.

We will describe this simulation in more technical details in Sections 4 and 5, but at this stage it is enough to know that this kind of simulation is possible today on a modern mobile terminal and can be used to greatly enhance many functions and features of the mobile terminal and various telecommunication applications.

4.1.3 Simulation tools for creating virtual 3D audio environments

When a sound source emits a sound wave into a spatial environment the listener will first be hit by the *direct sound* wave that propagates directly from the source to the listener. After that the listener will be hit by several sound waves, well separated in time and direction, that have been reflected once or a few times by neighboring walls, floor, ceiling and other reflective surfaces in the environment. These reflected sound waves are called *early reflections* and are illustrated in the reflectogram in the figure below as discrete and well separated reflections right after the direct sound.

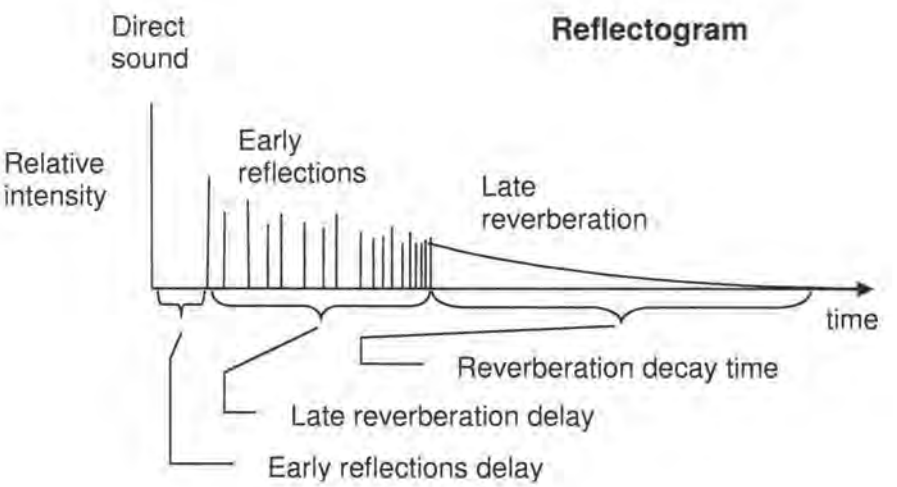


Figure 56 *A reflectogram showing the direct sound and all reflected sound waves originating from the same sound source reaching a listener in a room or an enclosed spatial environment.*

After a while the density of the reflected waves per unit of time will be so high that the distribution of reflections looks more like a continuous distribution than a discrete one. The intensity of those reflections will decrease exponentially with time. The time it takes for those reflections to attenuate by 60 dB is called the reverberation decay time. The sum of these dense reflections is called the *late reverberation* and is illustrated in the reflectogram in figure 56 as a continuous exponentially decaying distribution of reflections.

When simulating a 3D audio environment for a listener the three components described above (direct sound, early reflections, late reverberation) are usually simulated separately. The direct sound is important for giving the listener the sense of the direction to the sound source as well as for simulating the speed of the sound source. Simulating the early reflections properly gives the listener the ability to judge the distance to the sound source and it also enhances the externalization. The late reverberation mainly gives the listener the ability to perceive the acoustical characteristics of the simulated room, but it is also important for distance perception.

4.1.4 Direct sound simulation

ITD and HR filtering

The direct sound is the sound wave emitted by a sound source that propagates directly to the listener. Let us begin by looking at what happens when such a sound wave approaches a listener from a direction of arrival (DOA) that is not in the listener's median plane as illustrated in figure 57. Note that the DOA is specified in terms of the elevation and azimuth angles as illustrated in figure 54:

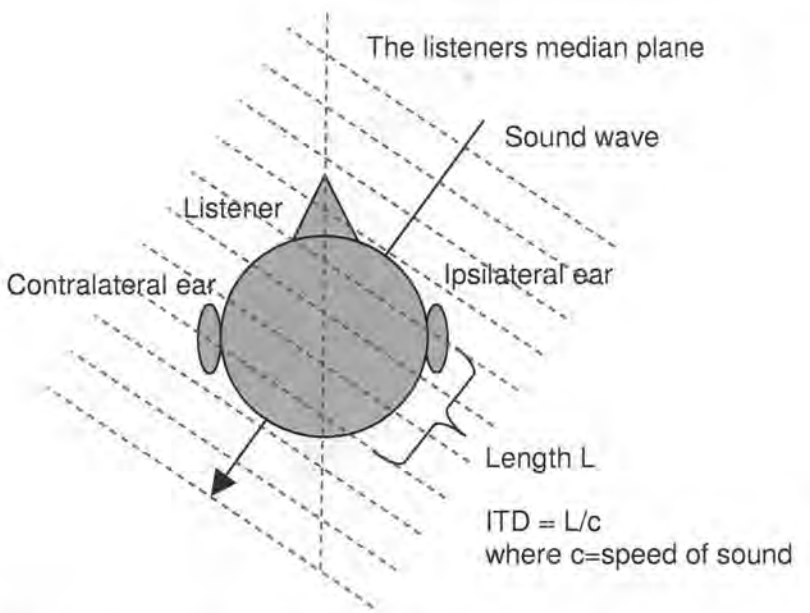


Figure 57 *When a sound wave approaches a listener from a direction that is not in the listener's median plane it will first hit the ipsilateral ear (closer to the wave front) and then the contralateral ear. The time difference is called the interaural time difference (ITD).*

First the sound wave will hit the ipsilateral ear (closer to the wave front) and then a few hundred micro seconds later it will hit the contralateral ear. The time difference, which is called the interaural time difference (ITD), is a function of the DOA. The ITD is also a function of the size of the head and does, therefore, vary between people with different head sizes.

On the way into the ear canals the sound wave is filtered by the upper torso, the head and the outer ears. This filtering, which for DOA:s that are not in the listener's median plane is different for the left and right ears, is called head related (HR) filtering. The HR filtering is also a function of the DOA and differs between people mostly due to the different sizes and shapes of the outer ears. Figure 58 shows a plot of the HR filters in the time domain for the DOA of elev = 0 degrees and azimuth = 45 degrees as measured from a human subject. Figure 59 shows the spectrum of the same HR filters.

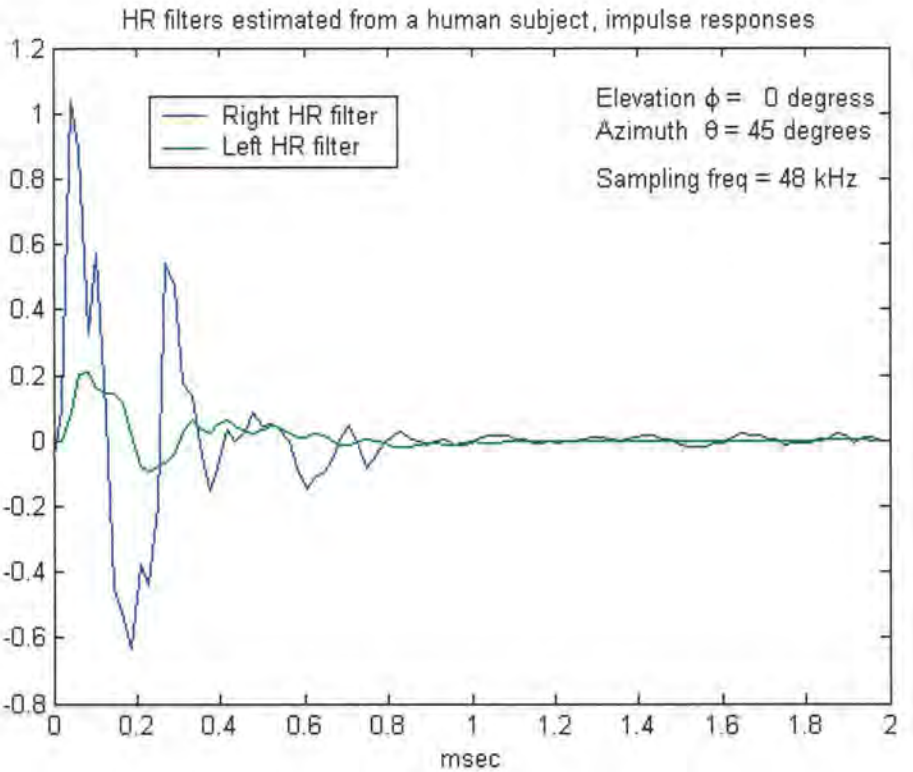


Figure 58 *Right and left head related (HR) filter impulse responses for the direction of arrival (DOA) of elev = 0 degrees and azimuth = 45 degrees, measured from a human subject at the sampling rate 48 kHz.*

By modeling both the ITD and the HR filters on a 2D grid of elevation and azimuth angles and using some form of interpolation for those DOA angles that are not on the sample grid it is possible to obtain the ITD and HR filters for any DOA angle. Having both the ITD and HR filters as functions of the DOA angle it is possible to generate the right and left ear signals of the listener for any DOA angle by the convolutions in Equation (57) .

$$\begin{aligned} e_R(n) &= h_R(n) ** s(n-\tau_R) \\ e_L(n) &= h_L(n) ** s(n-\tau_L) \end{aligned} \qquad (57)$$

where $h_R(n)$ and $h_L(n)$ are the impulse responses of the right and left HR filters and $s(n-\tau)$ denotes the sound source signal delayed by τ. τ is often a fraction of a sample and the time delay would need to be evaluated through resampling of the source signal.

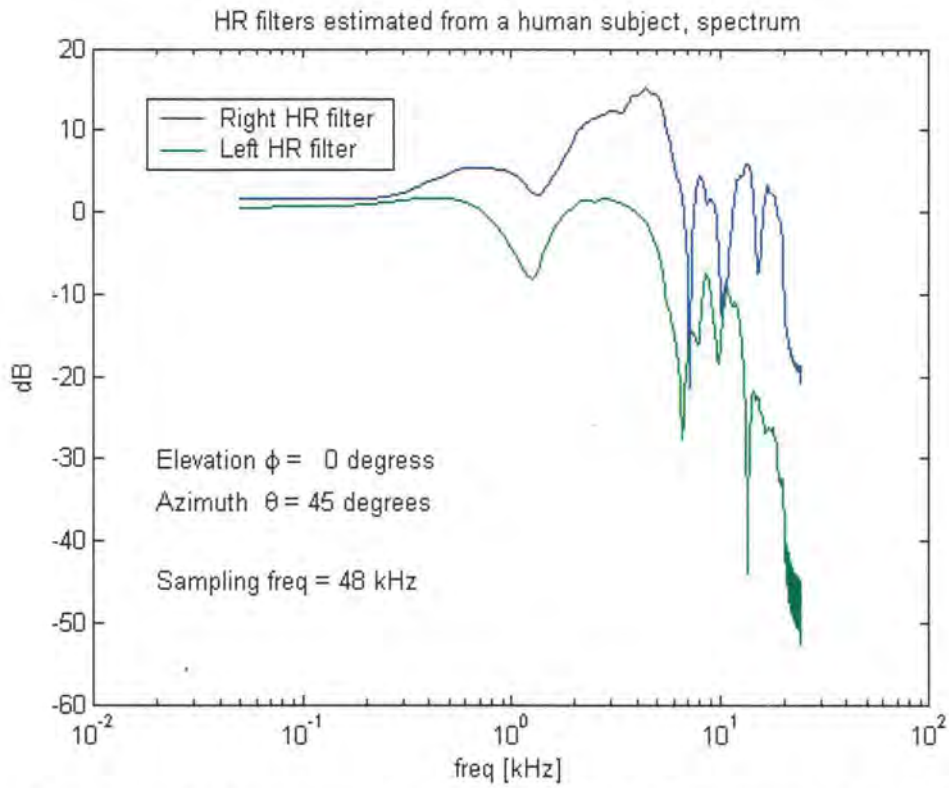


Figure 59 *Right and left head related (HR) filter spectrums for the direction of arrival (DOA) of elev = 0 degrees and azimuth = 45 degrees, measured from a human subject at the sampling rate 48 kHz.*

4.1.5 Early reflections simulation

When early reflections approach the listener each reflection will be filtered by the head and ears on its way to the eardrums and this can be simulated using head-related filters as described in the previous section. Before reaching the listener the reflections have bounced into different types of materials, which also cause filtering of the sound. The figure below shows a system that can be used to simulate the direct sound and early reflections.

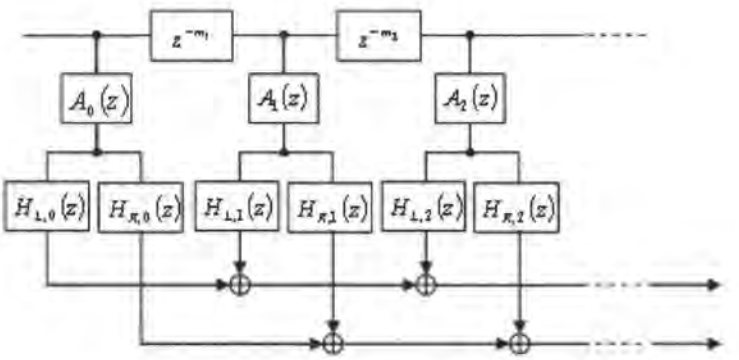


Figure 60 *HR-filters are applied to both the direct sound as well as to the early reflections.*

The HR-filters $H_{L,0}(z)$ and $H_{R,0}(z)$ are associated with the direct sound, which is given a gain $A_0(z)$, and the HR-filters $H_{L,1}(z)$, $H_{R,1}(z)$, $H_{L,2}(z)$, $H_{R,2}(z)$, . . . are associated with the early reflections that are given respective gains $A_1(z)$, $A_2(z)$, ... The gain filters for the reflections are used for simulating distance attenuation as well as the sound absorption that occurs during reflections. This generator can simulate early reverberation accurately, but applying HR-filters to the direct sound and all early reflections is computationally costly. In addition, the sound paths in a scene having moving sound sources change continually, which means that the corresponding HR-filters must be updated constantly and this is also computationally expensive.

Various attempts have been made to reduce the computational load imposed by the generator described above and Ericsson has one such solution.

4.1.6 Late reverberation simulation

In contrary to the early reverberation, the late reverberation is diffuse and its intensity is quite constant in a room, which means that the late reverberation is relatively independent of the positions of the sound sources and the listener. There is therefore no need of calculating the sound paths of the late reflections and instead it can be simulated in a more statistical way. Important characteristics of the late reverberation that need to be simulated properly are the reverberation time, the echo density and the modal density.

The reverberation time is defined as the time it takes for the room's impulse response to decay by 60 dB from its initial value and it depends on the size of the room as well as what type of materials its surfaces consist of. Typical values of reverberation time are a few hundred milliseconds for small rooms and several seconds for large rooms, such as concert halls and aircraft hangars. The reverberation time is frequency dependent, i.e. different frequencies decay differently, and this must also be simulated.

The echo density describes the density of the late reflections and it depends on how many reflective surfaces are present in the room that can give rise to reflections. For most rooms the echo density is high and a too low density often results in noisy artifacts, especially for impulsive sound sources.

The third parameter, the modal density, describes how dense the frequency peaks of the frequency response are. A low modal density will result in audible resonances and this can be true for some rooms, e.g. bathrooms, but most rooms have a high modal density resulting in a smoother frequency response.

The optimal solution for simulating late reverberation is to model two FIR-filters, one for each ear, as exponentially decaying Gaussian white noise with a spectral envelope that is formed by the geometry of the environment and the reflective properties of the walls in the environment. But since the reverberation time can be several seconds long this approach is not possible when real-time simulations are required and instead low complexity algorithms are used. Most common are reverberators based on comb and allpass filters or feedback delay networks.

Figure 61 shows Schroder's reverberator consisting of a comb filter and two allpass filters.

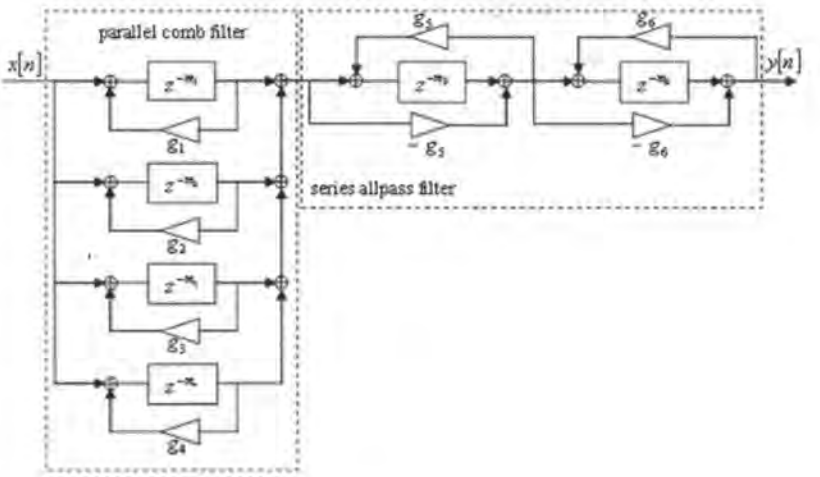


Figure 61 *Schroeder's reverberator with four parallel comb filters in series with two allpass filters.*

The second type of reverberator, feedback delay networks (FDN), can be seen as an extension to Schroeder's parallel comb filter structure. Instead of directing the output of the comb filter to only its input, the output is also directed to all other comb filters' inputs via a feedback matrix. Stautner and Puckette [42] proposed a four-channel reverberator consisting of four delay lines with a feedback matrix resulting in a structure that was capable of much higher echo densities than Schroeder's reverberator.

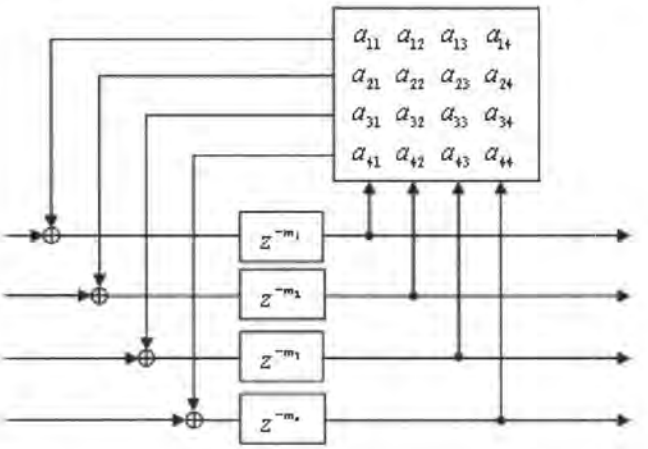


Figure 62 *A feedback delay network with four delay lines.*

The advantage of the FDN compared to Moorer's reverberator is that the echo density increases exponentially over time, while it is constant for Moorer. The drawback, however, is that the FDN is more computationally expensive than Moorer's and if the similar complexity should be obtained with both methods, the dimension of the feedback matrix must be lower than the number of comb filters. This means that the beginning of the FDN's impulse response will be less dense than Moorer's, but after a while the echo density will build up and become denser.

Several improvements of the above described methods have been developed through the years and Ericsson has such improved solutions implemented.

4.1.7 3D audio API on EMP's mobile platforms

The 3D audio API on EMP's platform is designed to integrate the 3D audio effects into the general structure of the audio system so that they can make efficient use of all the functionality provided by the audio system and be mixed together with other audio effects. To achieve this, the API is divided into two layers, the ***3D Audio Scene Management Layer*** and the ***Audio System Layer*** as illustrated in figure 63.

In the 3D Audio Scene Management layer the ***3D Audio World*** object is the object used to control the 3D audio rendering engine. It is created by a method from the ***Audio System Effect Interface*** and has its own methods for creating the ***3D Audio Listener*** object, the ***3D Audio Source*** objects and the ***3D Audio Effect*** objects. The Audio Source object represents the audio source in the 3D space with methods for moving the object in the space, while the Audio Effect object is the object where the actual audio effect is implemented and is part of the audio system.

The platform supports 4 simultaneous 3D Audio Effect objects, while the number of 3D Audio Source objects can be much higher. A 3D Audio Source object can only be connected to one 3D Audio Effect object at any given time. A game designer that is using more than four audio sources can use this separation of the 3D Audio Source object and the 3D Audio Effect object so that at any given time she only connects the 4 most important audio sources in the scene to the available 3D Audio Effect objects.

The 3D audio renderer supports two modes for updating the audio effect parameters used by the audio system, the ***immediate*** and the ***deferred*** mode. In the immediate mode all audio effect parameters are updated immediately after each change of the geometric parameters of the 3D Audio Listener or 3D Audio Source objects, while in the deferred mode the update of the audio effect parameters is deferred until a ***commit*** method is specifically called. The deferred mode of operation is the most natural and efficient to use in a game or some other 3D animation as it allows the game developer to update the complete geometric scene before the audio effect parameters are updated. The immediate mode of operation, on the other hand, leads to a great many and unnecessary updates of the audio effect parameters, each of which requires a great deal of computations and data transfers. The 3D Audio World object has a method for setting this mode of operation as well as the commit method.

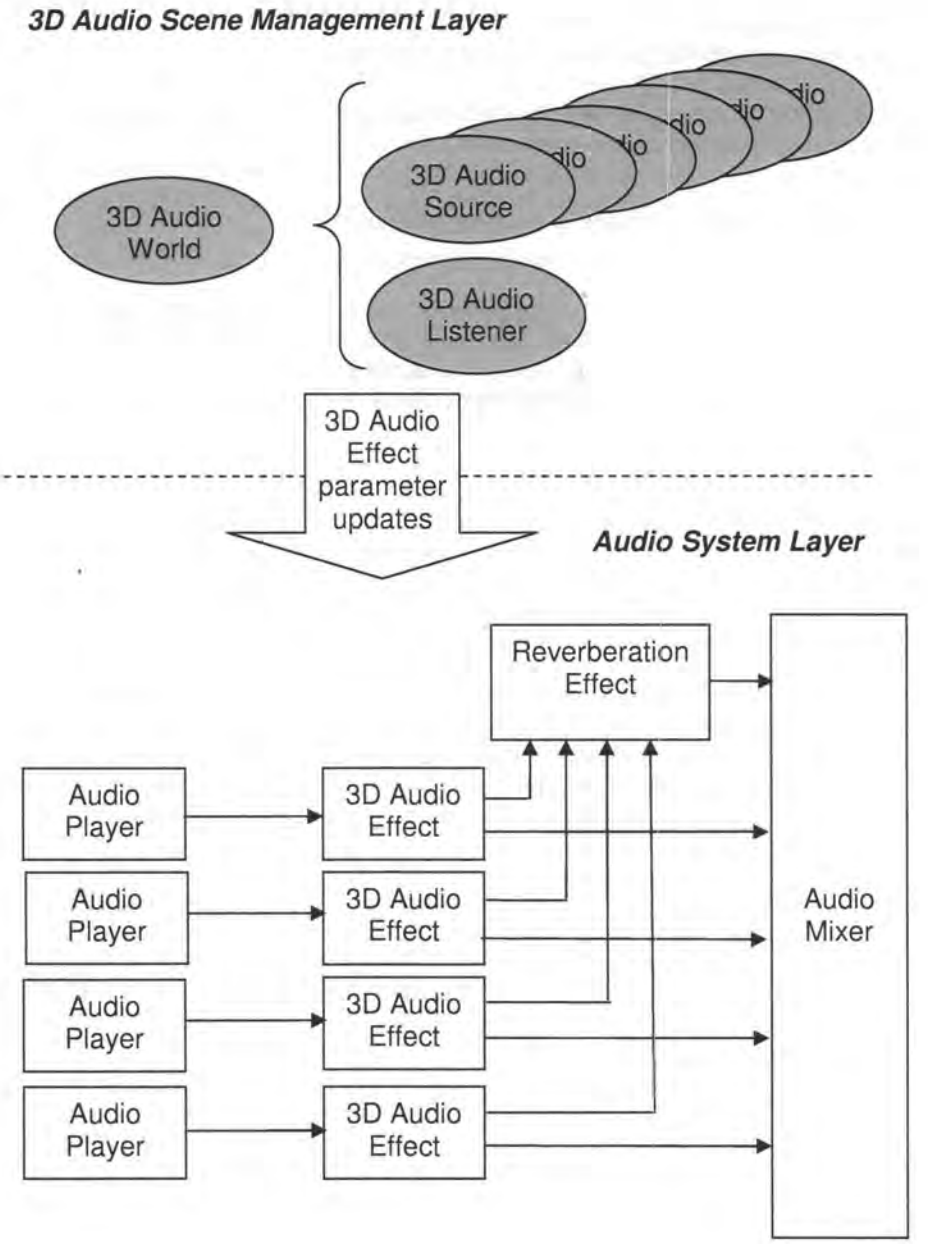


Figure 63 *A block diagram showing the split of the 3D audio rendering engine into a scene management layer and the audio system layer.*

For the 3D Audio Source objects there are also methods for enabling, disabling and setting the distance and directional gains, the Doppler effect and the reverberation level (the level of the feed sent to the reverberation unit).

For the 3D Audio Listener there are methods for setting the head and ear sizes of the listener that control the ITD and HR filters used in the rendering. Three sizes are supported small, medium and large. The ITD functions and HR filters used come from Ericsson Research and are derived from a large database of ITD functions and HR filters measured from over 50 people.

Currently there is no implementation of the early reverberation, but the late reverberation is a Moorer type of an implementation. The Reverberation Effect is available as one of the effects supported by the audio system as it can also be used on other audio signals than those delivered by the 3D Audio Effect. As such it is managed through the Audio System Effect Interface. The parameters that can be set for the Reverberation Effect at this time are summarized in Table 1.

The block diagram shown in the Audio System Layer in figure 63 illustrates a relatively simple audio network with four 3D Audio Effect blocks. Each 3D Audio Effect block obtains its input directly from a player and has outputs to both the Reverberation Effect block and the Audio Mixer. All audio effects other than the 3D Audio Effect are created through the Audio System Effect Interface, while the interconnection from a player to an audio effect and from an audio effect to another audio effect is managed through the Audio System Interface.

More details on how to actually set up a 3D audio network and get it working can be found in the OPA documentation on Audio Management, Audio Effect Management and Player Management.

Ericsson was heavily involved in the development of the audio components of the Java standard JSR-234 and the EMP 3D audio rendering engine is closely coupled to that standard with regard to functionality and architecture.

Table 1. Late reverberation user parameters.	
Wet and dry levels	Each audio effect outputs a linear sum of the input signal (dry part) and the effect signal (wet part). The dB levels of these two components can be set. In 3D audio the dry and wet components are handled by the 3D audio rendering engine, which requires the dry level to be set to the lowest possible dB level and the wet level should be set to 0 dB.
Early reflections delay	The delay from the direct sound to the first early reflection as illustrated in Figure 56.
Reverberation delay	The delay from the first early reflection to the beginning of the late reverberation as illustrated in Figure 56.
Reverberation decay time	The time it takes for the reverb to decay from its initial level by -60 dB.
Reverberation frequency decay time ratio .	The value of the high frequency decay time in percent of the low frequency decay time.
Reverberation level	The level of the dense late reverb relative the reflection level in dB. 0 dB gives a reverb tail with roughly the same energy as the input signal. It changes the sound of the reverb, and should not be used to adjust the volume of the reverb.
Room type	The supported room types are: Default, Generic, Padded cell, Room, Small room, Medium room, Large room, Bathroom, Living room, Stone room, Stone corridor, Carpeted hallway, Hallway, Medium hall, Large hall, Auditorium, Concert hall, Cave, Arena, Hangar, Alley, Forest, City, Mountains, Quarry, Plain, Parking lot, Sewer pipe, Underwater, Musical plate.

4.2 3D audio integration

4.2.1 Architecture of the head tracking application

In order to make the head tracking software accessible by other applications, the head tracking solution should be complied as a library. The advantage of this is that other applications can use the functionalities provided in the library independent of which application it runs from.

Libraries have a gateway for communication between the application and the library called the *Application Programming Interface* (API). API defines of a set of routines that can be called from outside. It allows an application program to access the functionality provided in a library that is to request a service from library.

Figure 64 *API acts as a door gate that allows an application request a service from a library*

In this project however, the head tracking library is programmed and compiled in C. A program written in C runs fast because it complies closed with native machine code. In addition, unless a platform specific API is used, the C library it can be compiled on almost any platform. Moreover, the Wii remote driver library **wiiuse** is also in C.

Hence, the head tracking library receives 2-dimensional data from the Wii remote through the Wii remote driver, wiiuse. The data has a range of 1024 by 768 pixels. The task of library is to convert these data to 3D position and orientation of the head. In order to updates the scenes, the 3D audio rendering engine requires a set of 3D position for the head position and a front and up vector as the head orientation.

The structure of head tracking application can be summarized as in the scheme in figure 65:

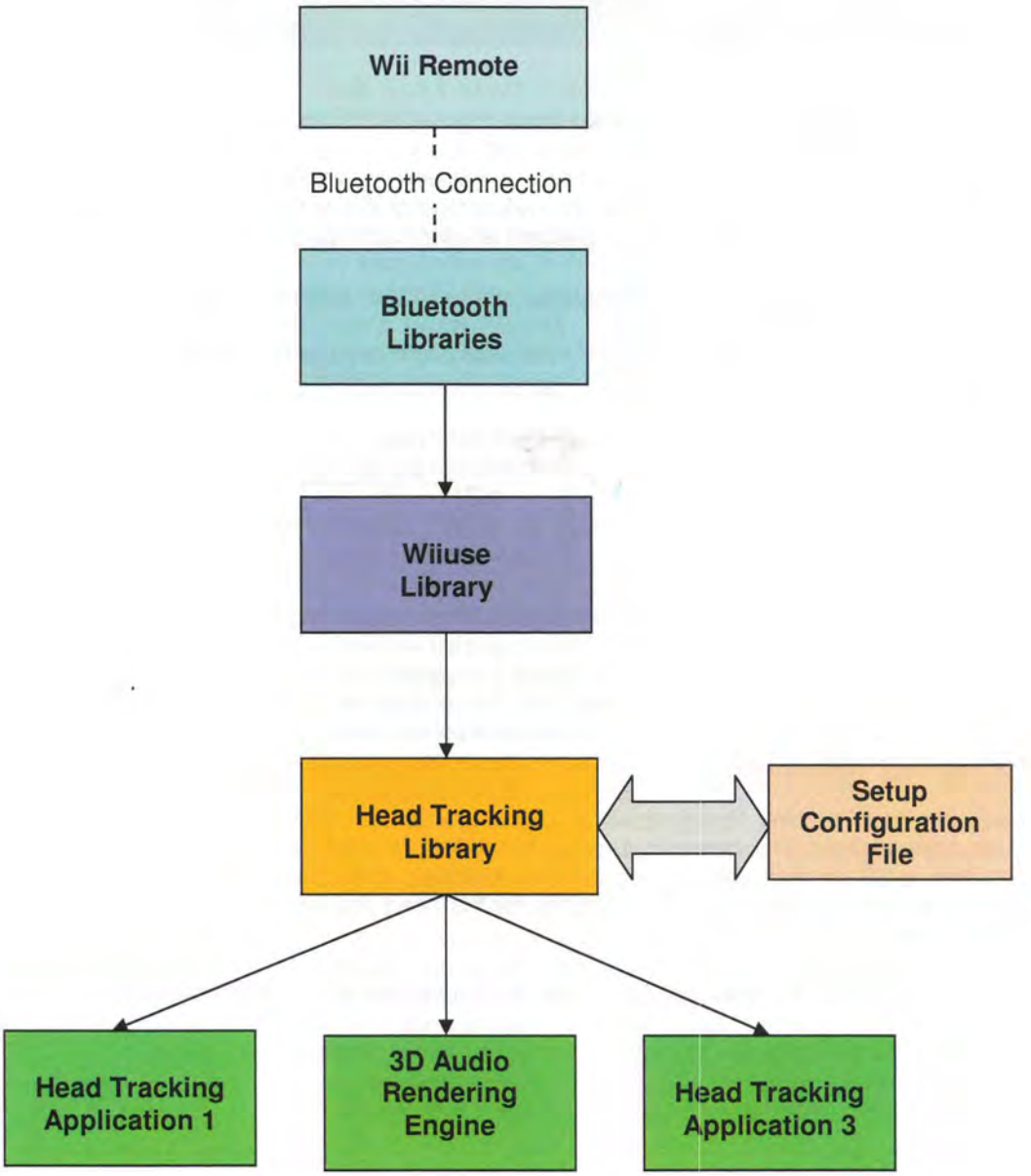


Figure 65 *The architecture of head tracking library used in the head tracking application*

4.2.2 Realizing the library

4.2.2.1 Library interface

Generally speaking, the head tracking applications are interested in the position of the head and they want to receive the updated information as many times as it is possible for the tracking device to supply the information. Our head tracking application, the 3D audio rendering enginer, receives the information about position and orientation of the head during tracking from the Wii remote and the head tracking algorithm. Orientation is supplied in terms of front and up vector. The updates are delivered about every 10 millisecond.

4.2.2.2 Library external functions

An external function is linked to the head tracking library to access the functionalities inside the library. This function contains an initialization function *void InitPoseEst ()* that initializes the library with the initial parameters required for head pose evaluation. This is normally done once in the beginning of the program call and the values are used during the tracking.

In addition, the external function includes a function called *void GetPosePosOrient ()* which is the main function responsible for calculating the head pose. This function evaluates the position of the rigid body as well as the front and up vector associated with rigid body movement in three-dimensional space. As soon as the new 2D data are available, *GetPosePosOrient()* is called which in turn evaluates the output parameters and override the parameters in a data structure that is shared by the 3D Audio Engine. The 3D Audio engine collects the new data and updates the audio scenes associated with the listener location in 3D space.

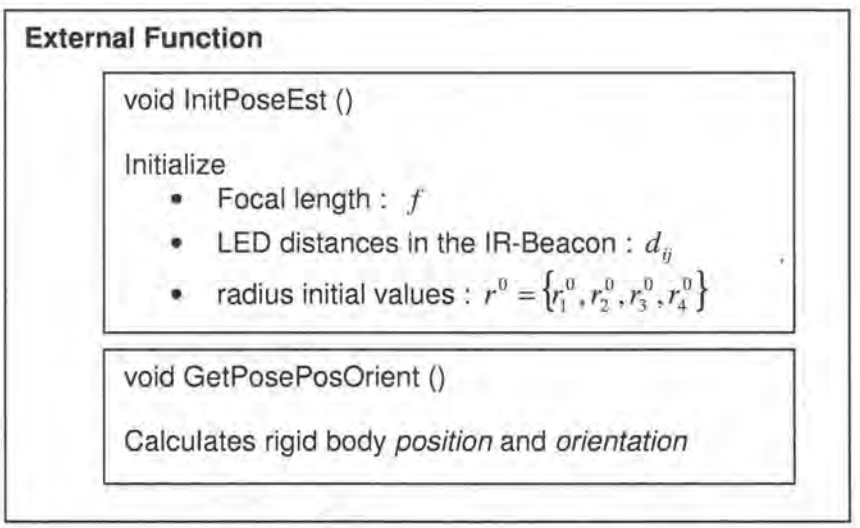


Figure 66 *The External function linked to the head tracking library*

4.2.2.3 Initialization

Initialization is done by clicking the Play Button on the graphical window. This activates the WiiRemoteThread which keeps polling the Wii remote in a regular interval. In addition, this call initializes the wiiuse library. Then the configuration file is read by the library which contains the information about camera focal length, distance between LEDs in the IR-Beacon and radius initial values. The parameters are used to initialize the *InitPoseEst()* function. The next call is to connect to the Wii remote. When the connection is established, the IR tracking has to be enabled. The head tracking application enables the IR tracking by setting the flag START IR_TRACKING which is activated by pressing up button in the wii remote. This causes the library to call *wiiuse_set_ir()* function which in turn activates the IR tracking. A typical sequence of function calls for the head tracking application is displayed in the sequence diagram in figure 67.

4.2.2.4 Polling

In order to get the data from the wiiuse library, a polling function is required to pull the library regularly. This is implemented by placing the polling function inside a while(1) loop which is activated/deactivated by setting START_IR_TRACK /STOP_IR_TRACK flags. This keeps polling the wiiuse in a regular interval. The polling part continues for as long as the application runs (see figure 67):

Polling Loop

```
while(1){ if(START_IR_TRACK=TRUE)
Start Tracking;

if(START_IR_TRACK=TRUE)
Stop Tracking;

wiiuse_poll();}
```

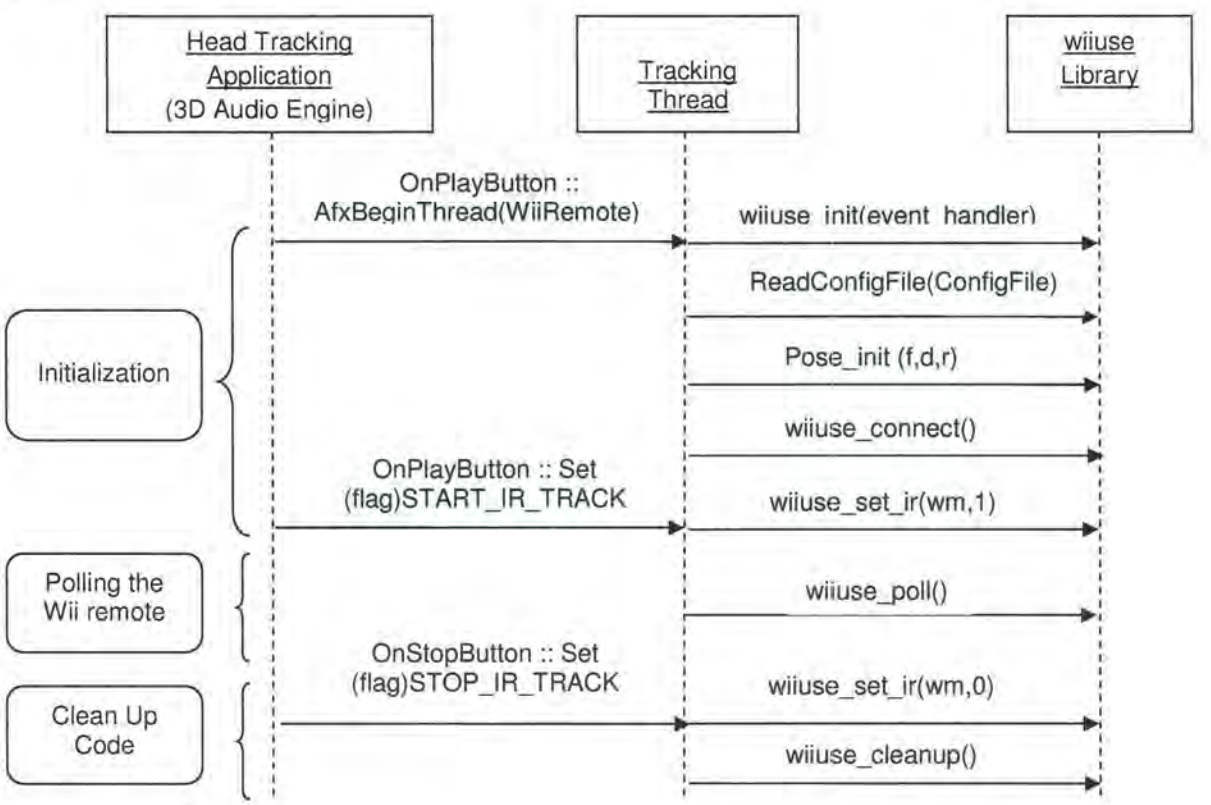


***Figure 67** A sequences of function calls for the head tracking application*

4.2.2.5 Threading

The program is a *multithreaded* code in which each thread is responsible to perform certain tasks in a concurrent manner with respect to the other threads. These threads may share same memory storage.

One of these threads is the *WiiRemoteThread* which communicates with the Wii remote and collects the raw IR data from the device about every 10 msec. These raw data are then interpreted as pose data by the pose estimation part and the result are stored in a memory storage namely *ListenerPose Data*. ListenerPose Data are global variable that are shared by two threads. In order to avoid the simultaneous use of common data **mutual exclusion**[13] is used. Mutex acts as like a LOCK to the data memory and protects the data from being corrupted because of mutual concurrent use, or Thread TimeOut and so forth. The key to this lock is only available to the threads sharing the same memory storage, for the case of Listener PoseData the *WiiRemoteThread* and the *SceneUpdateThread*.

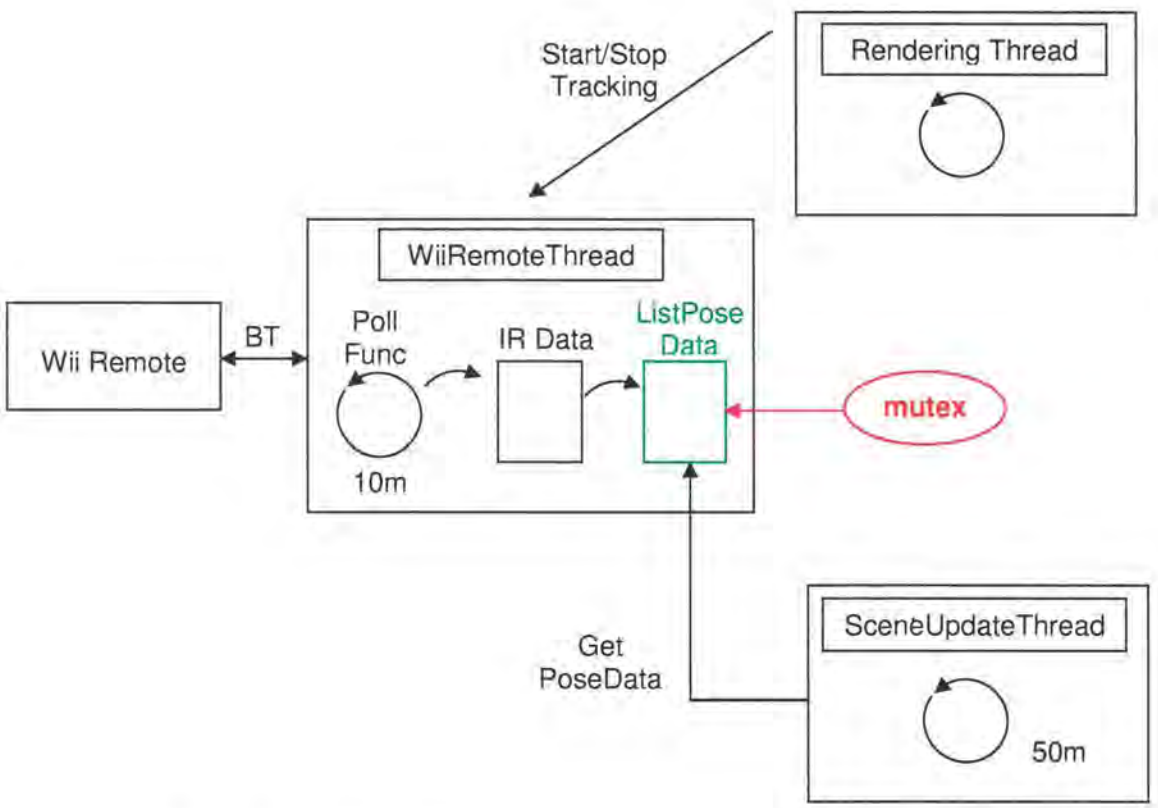


Figure 68 *Threads involved in the head tracked 3D audio rendering engine*

[13] mutex

5 Results

5.1 Evaluation based on simulation

To analyze the performance of the pose estimation in a controlled manner, we have simulated the image plane data from the camera for a random walk movement of the rigid body. This analysis aims to compare the estimated movement of the rigid body with the actual random walk movement to see how well the algorithms are tracking the actual movement.

So a known rigid body (e.g. human head) centered at the origin is rotated and translated into arbitrary points in the space to give the transformed rigid body. This happens when the listener moves in the space. Each pair of transformation parameters (R_i, T_i) gives a set of projections on the camera image plane defined by the camera projection equations (see section The pinhole camera model). These are the raw data captured by camera. The camera quantization effect is considered here to obtain a realistic model of the camera capture. Camera quantization is due to 8x subpixel analysis it uses to reach a high interpolated resolution. Thus a uniform quantization of level 8 (and quantizer step size 8) is applied to raw data to model the quantization effect of camera. Quantization is the primary source of inaccuracy in the measurements and therefore it is important to obtain an insight about its effect on measurement results. By taking this procedure, we can simulate the raw data and use our head tracking algorithm on simulated data.

In the next stage, we do the reverse manner from 2D world to 3D world to obtain the 3D coordinates of rigid the body target points from the 2D data by means of the pose algorithm. All the simulations are done in MATLAB and MATLAB plays a key role in the simulation of the algorithm.

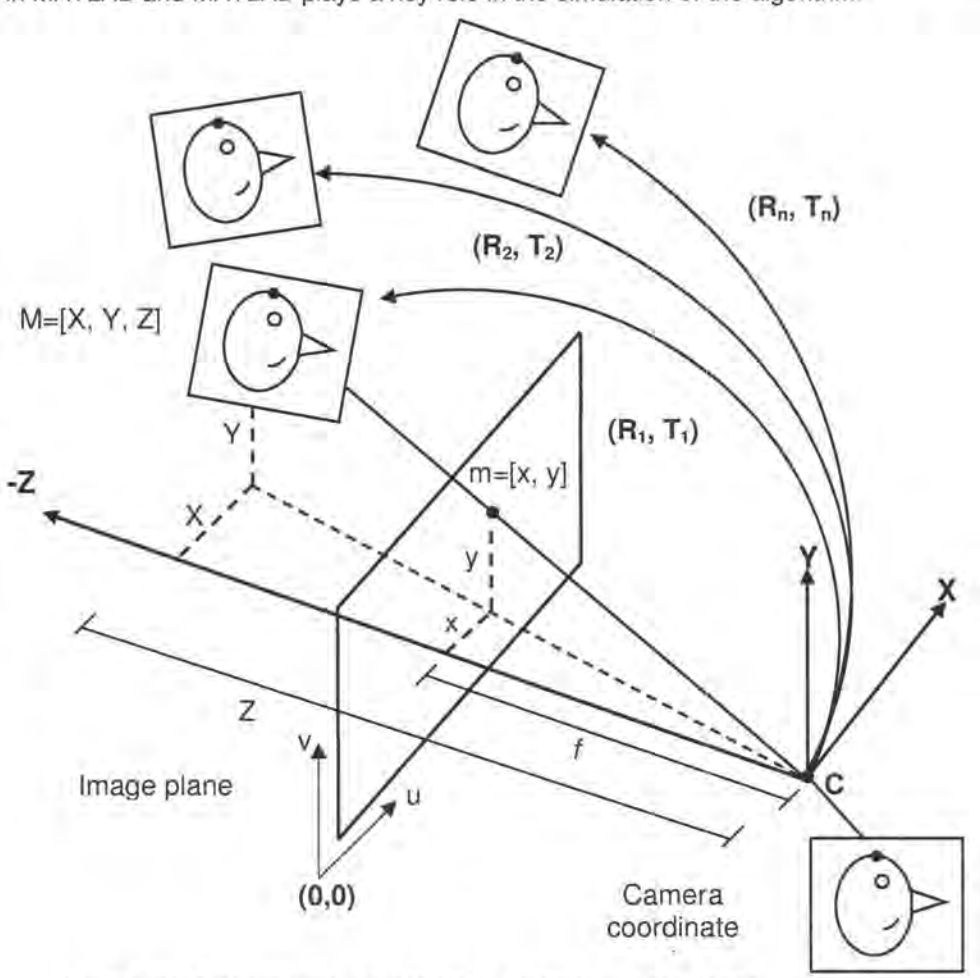


Figure 69 *Randomly movement of the rigid body in the space*

The two main procedures are described below:

5.1.1 Simulation of the camera capture

This is achieved by randomly movement of the rigid body (translation and rotation) into arbitrary points in the space followed by the simulation of the captured data.

The following figure illustrates the above procedures:

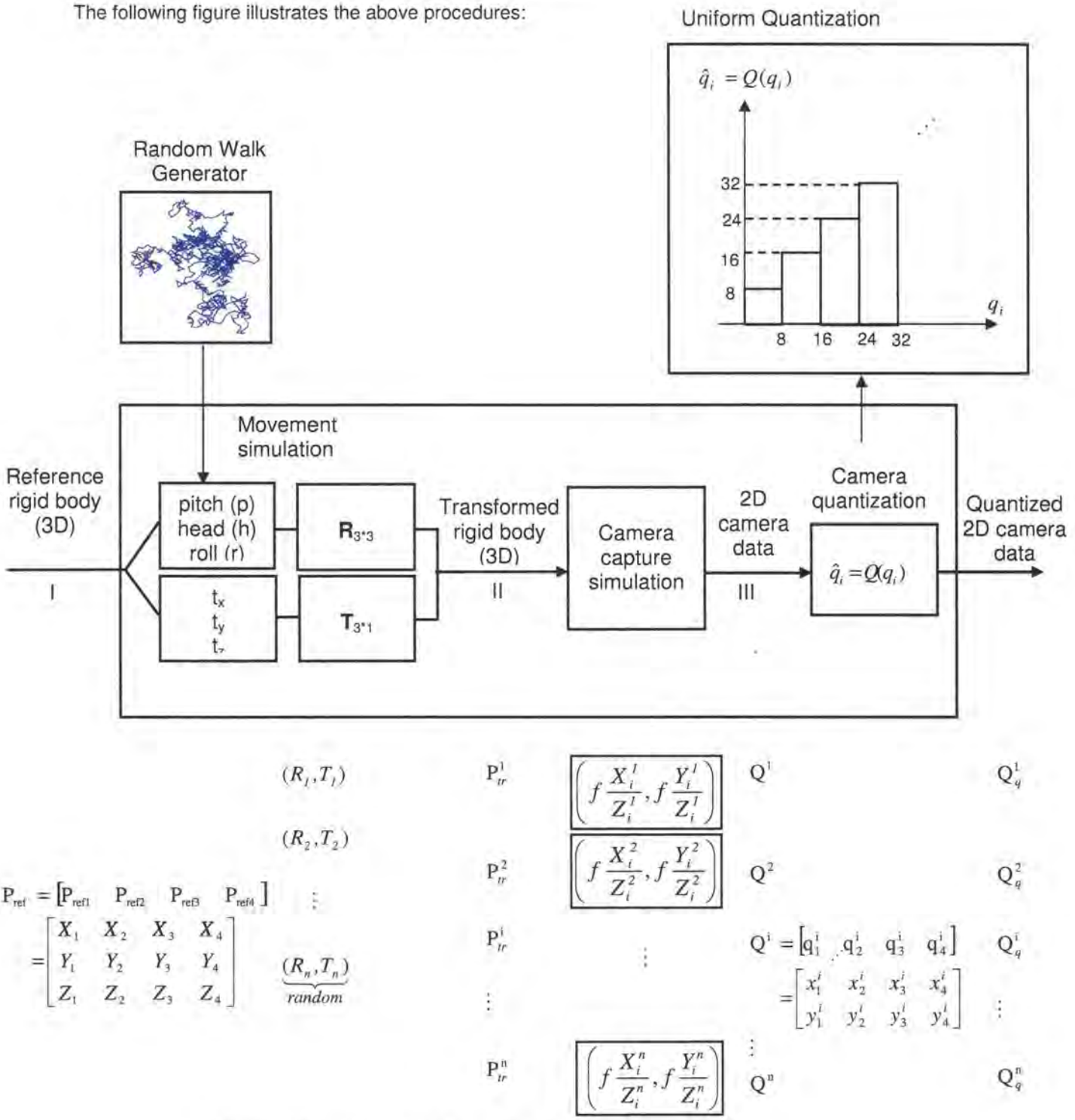


Figure 70 *Simulation part 1: simulation of the camera capture*

5.1.2 Evaluation of pose based on simulated capture data

The entire process of estimating the angles and translations from the image plane data is illustrated below. The quantization function is a uniform quantization of the image data to points that are integer multiple of 8 and is placed due to the uncertainty about real resolution of the camera.

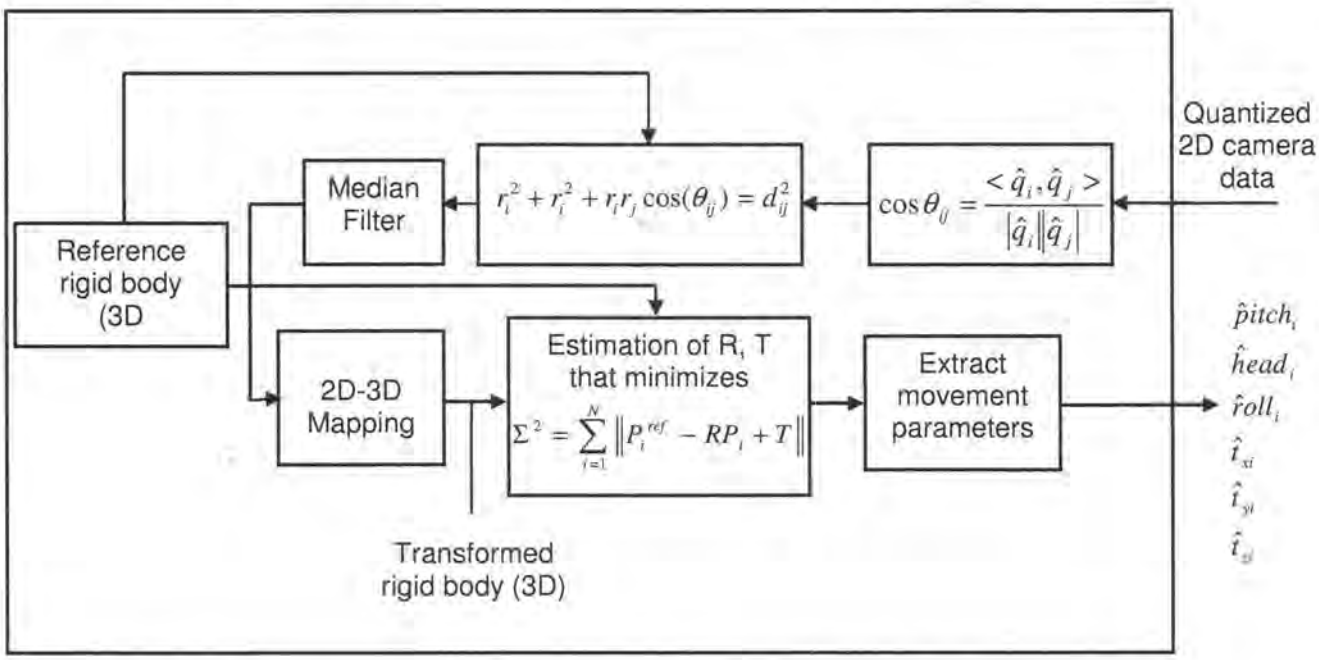


Figure 72 *Simulation part 2: evaluation of pose based on simulated capture data*

5.1.3 Compare the estimated results with real ones

The plots of estimated angles and translations are provided in the following figures. In the top figures, the black curves show the real data used to move the rigid body in the space (step 1 of the simulation). The blue curves show the estimated parameters from the image plane data (step 2 of the simulation). The black curves show the estimated parameters when smoothing filter is considered.

The bottom figures, however, depict the error composed of difference between the ideal parameters and estimated ones when filtering is considered. Different types of plots are used to provide a statistic insight about the algorithm performance in presence of quantization noise. For the sake of comparison the plots are provided with two levels of quantization (low and high).

In most cases, the error is relatively proportional to the absolute value of parameters and shows the deviation of less than 5% from the ideal results which are by large due to quantization noise considered in the input of the equations. The last two figures (figure 75 and 76) show the same plots with 1-level of quantization. The level of error in this case is by large extent decreased which strengthens the hypothesis that inaccuracies are most related to quantization noise.

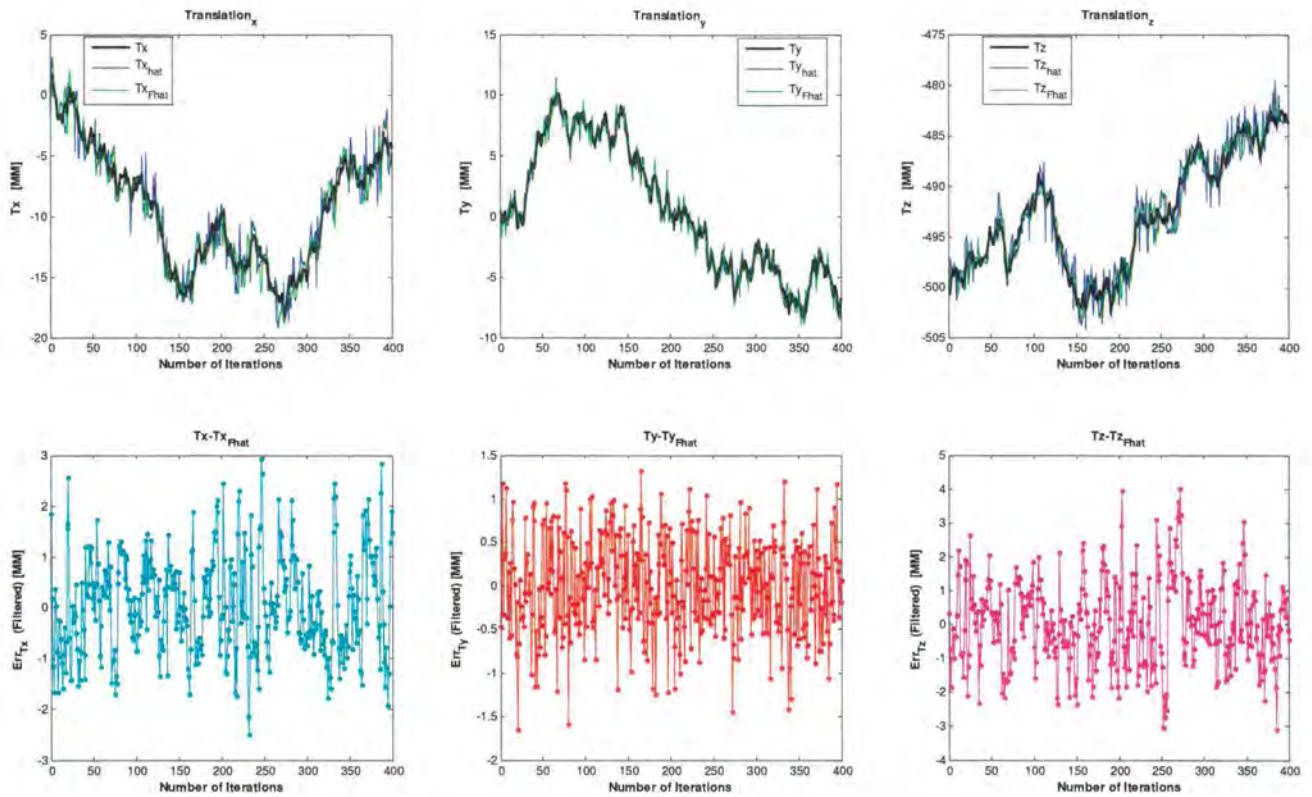


Figure 73 *Translation plots (3DOF) with 8-level of quantization*

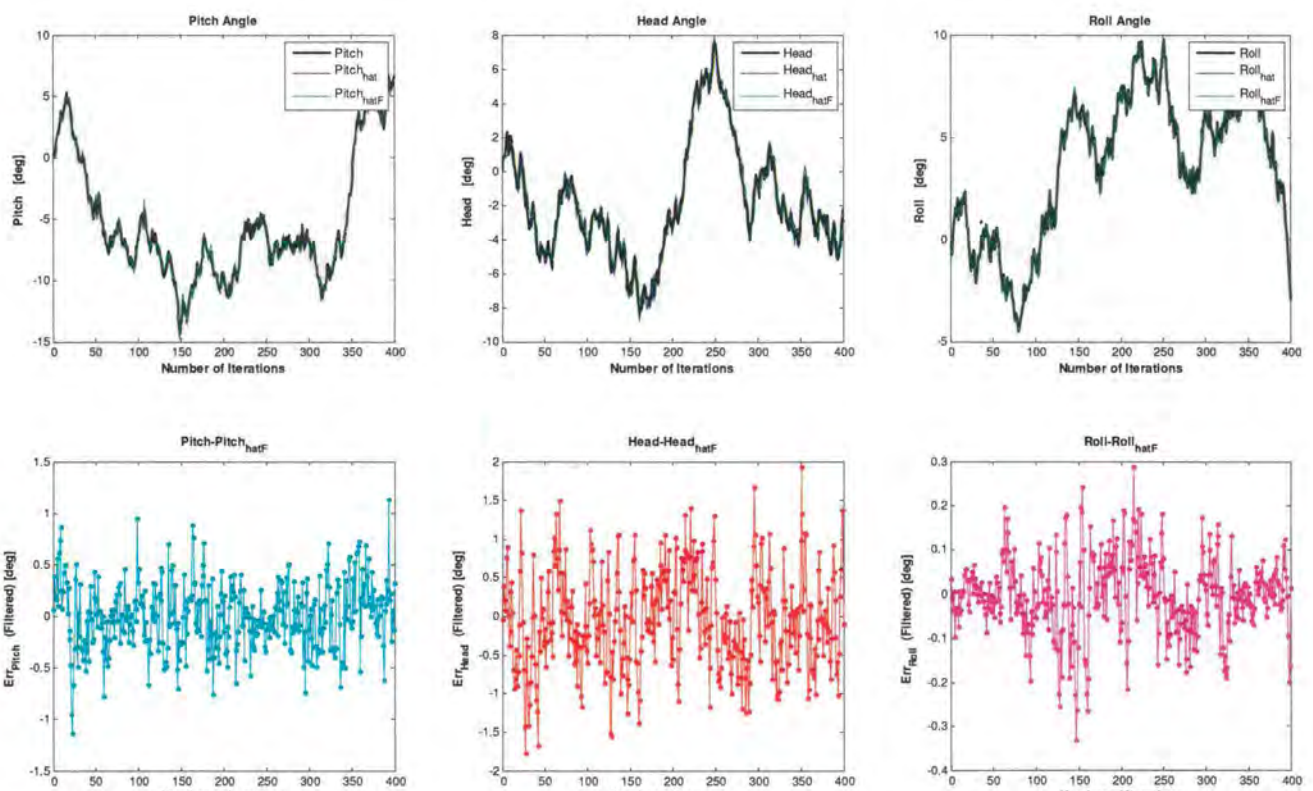


Figure 74 *Rotational angles plots (3DOF) with 8-level of quantization*

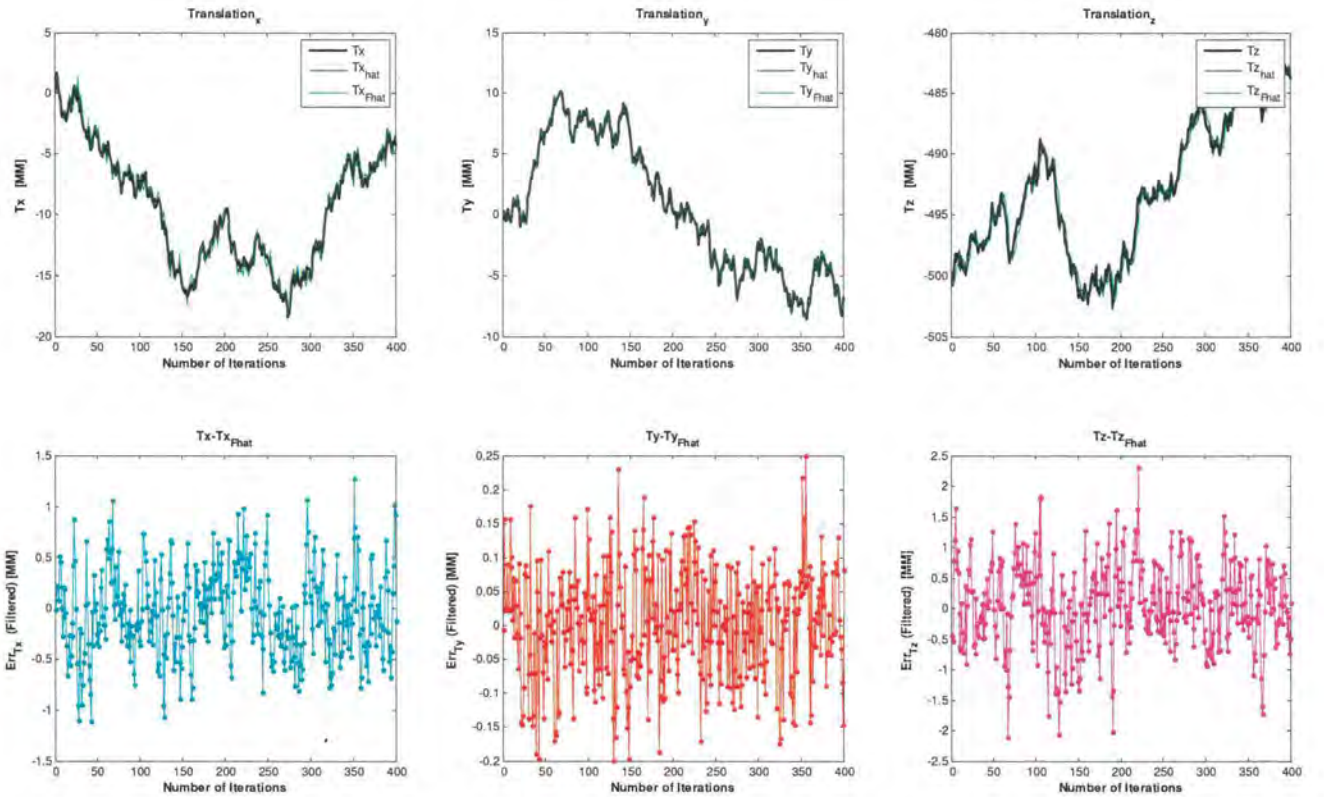


Figure 75 *Translation plots (3DOF) with 1-level of quantization*

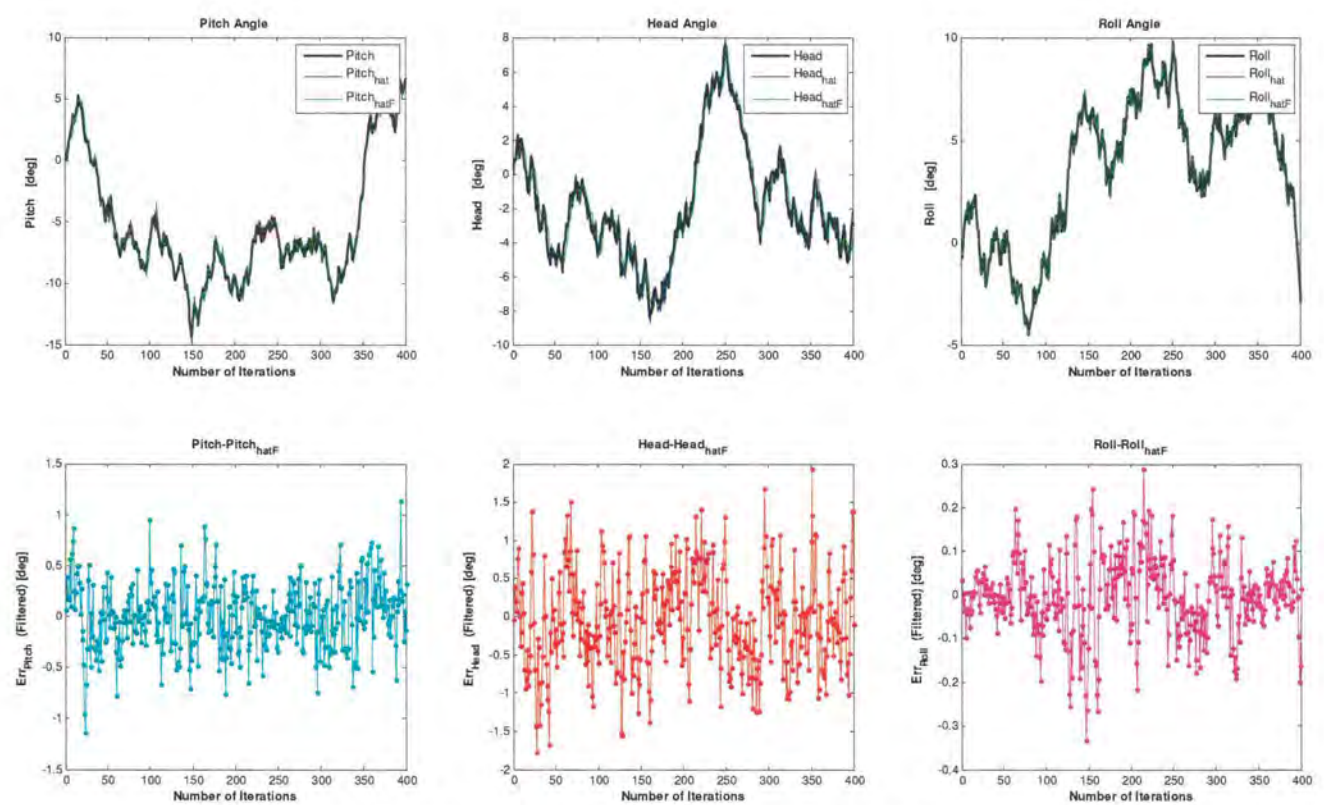


Figure 76 *Rotational angles plots (3DOF) with 1-level of quantization*

5.2 Sensitivity analysis

The aim of the sensitivity analysis is to investigate how sensitive the solution is to:

1. Starting value of the iterative algorithm
2. Rigid body configuration (relative position of the diodes)

For a realistic simulation, the sensitivity is analyzed in presence of quantization noise.

Probability distribution functions and scatter plots are provided in this section for analysis. A scatter plots is a mathematical function to display values for two variables for a set of data.

5.2.1 Sensitivity to initial values

During the iterations, if the evaluated results are not fed back into the algorithm as the next initial guess, the algorithm may diverge at some occasions (compare figure 78 and 79). Therefore, feedback of previously estimated radius parameters as initial guesses in current iterations is strongly recommended. This may also have a positive effect on convergence speed.

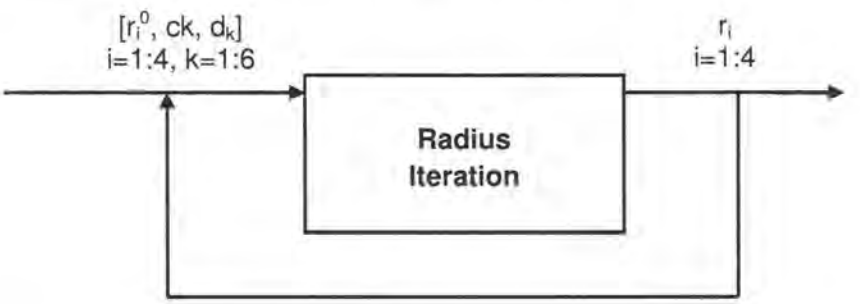


Figure 77 *Feedback of the estimated radius parameters an initial guesses in current iteration*

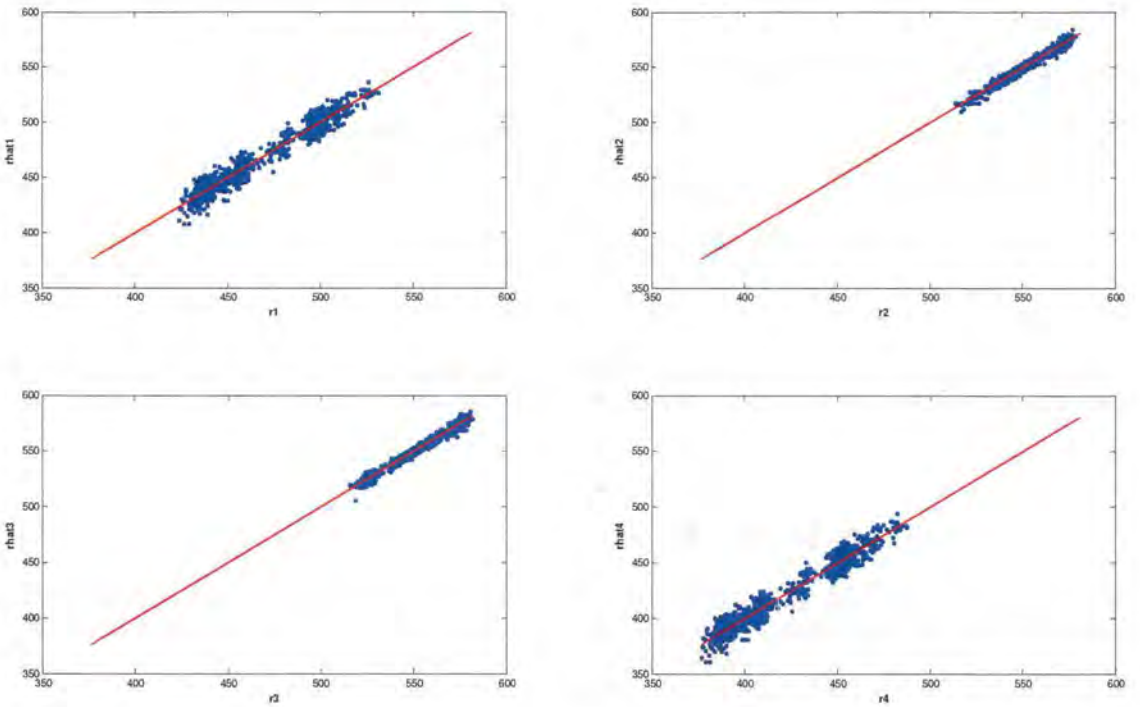


Figure 78 *Scatter plots for radius evaluation WITH feedback*

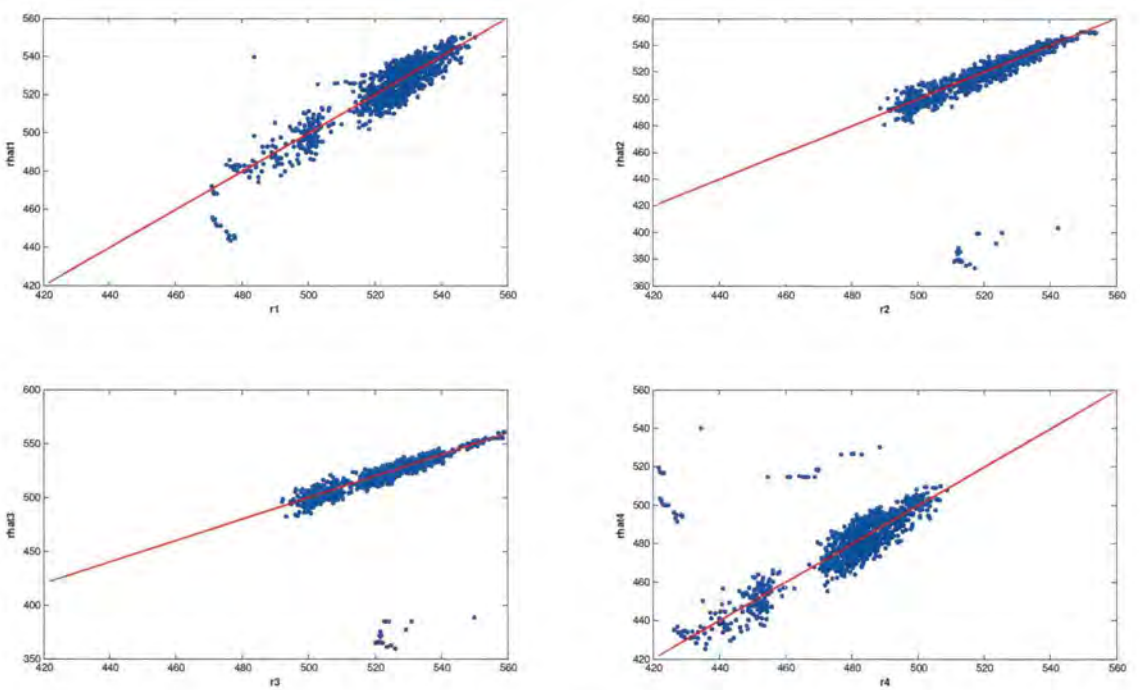


Figure 79 *Scatter plots for radius evaluation WITHOUT feedback. Divergence is noticeable at some occasions.*

5.2.2 Sensitivity to coplanar arrangement

The four LEDs on the rigid body have to form a non-planar arrangement. Coplanarity of the diodes leads to divergence because of the vanishing gradient and therefore is strongly prohibited. Figure 80 shows one example of divergence due to coplanarity. That is why three diodes are more susceptible to divergence because three diodes always form a plane. With four diodes however, the coplanarity problem could be alleviated by simply mounting one of the diodes higher than the others (e.g. 3-4 cm above).

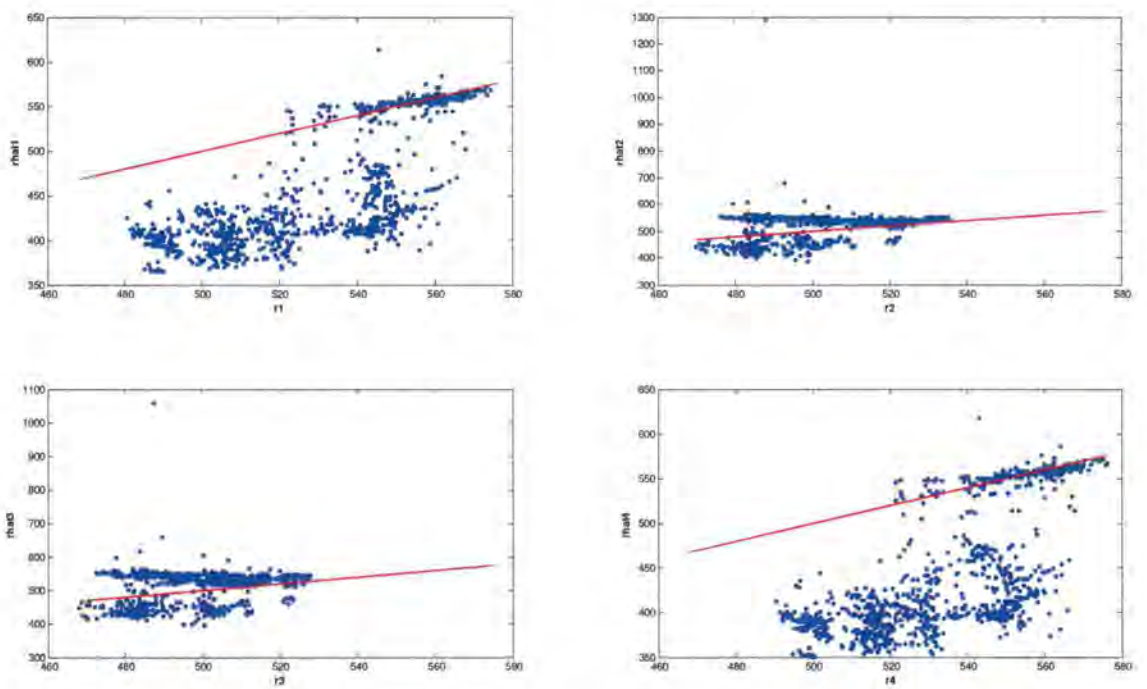


Figure 80 *Scatter plots for radius evaluation for a coplanar arrangement. Divergence is noticeable.*

5.3 Evaluation based on recorded data

In addition to simulation, the head tracking solution was tested with real data. We started doing this in MATLAB, because it is so easy to work in MATLAB and verify the results with different types of plots that the software can provide.

Accordingly, in a real experiment, the Wii camera was used to track the rigid body movement in the space. The data were saved in a file to be used for the head tracking algorithm. The data were then imported to MATLAB as the input image plane data and the head tracking algorithm was tested on real-data to analyze the movement calculated by the algorithm.

Different plots such as radius, translation and orientation were provided for this purpose and are shown in the following figures. In addition, an airplane flight simulator was created to provide a mechanism to visualize the movement through animation. The airplane in the animation is able to move/rotate in all 6DOF.

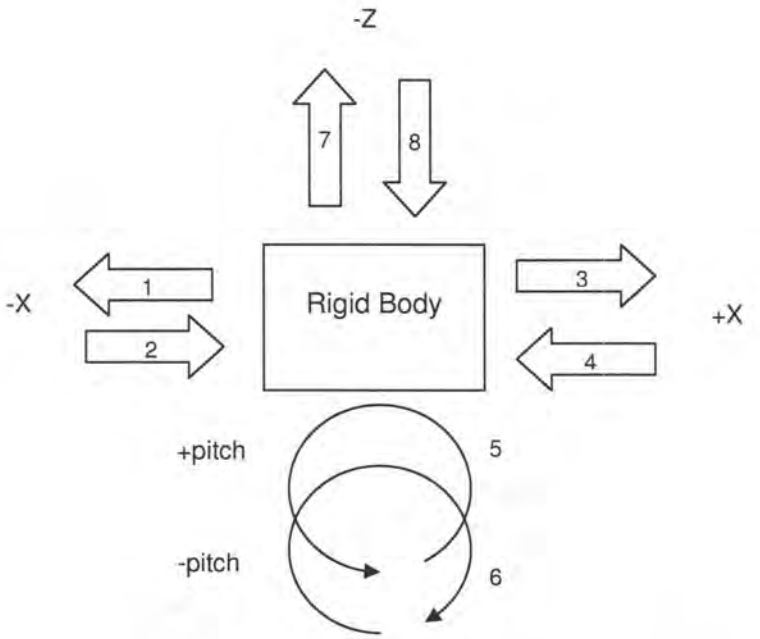


Figure 81 *An arbitrary movement of the rigid body during an experiment*

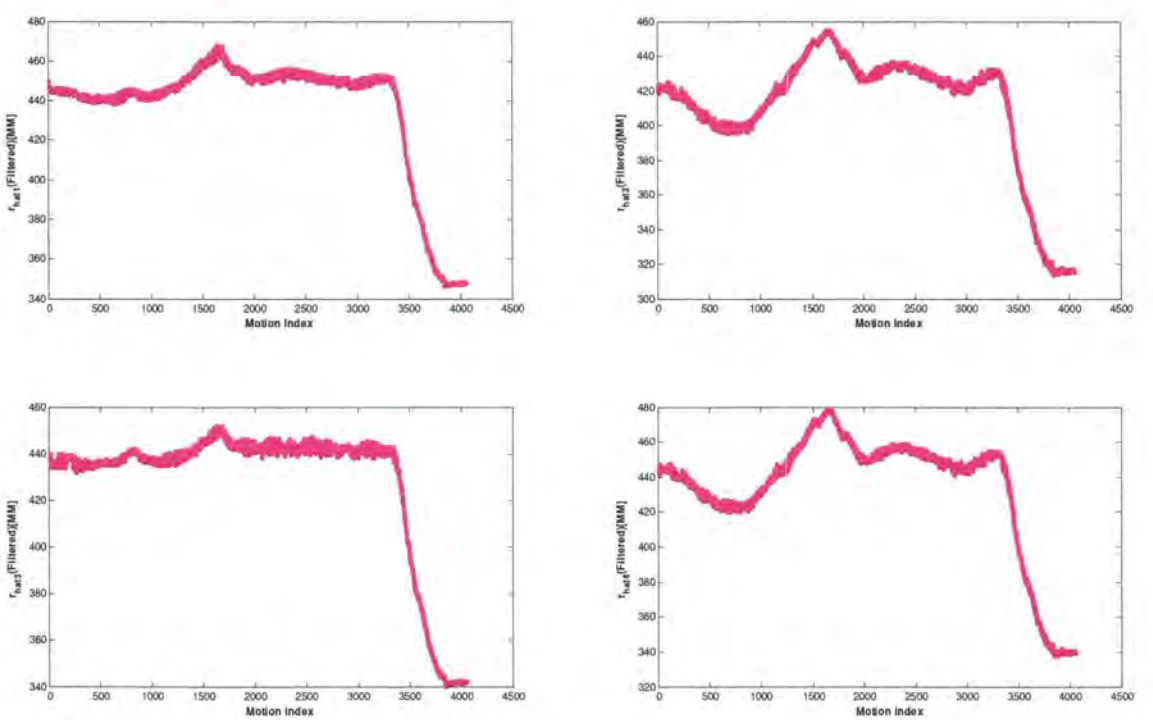


Figure 82 *Radius results for each target point*

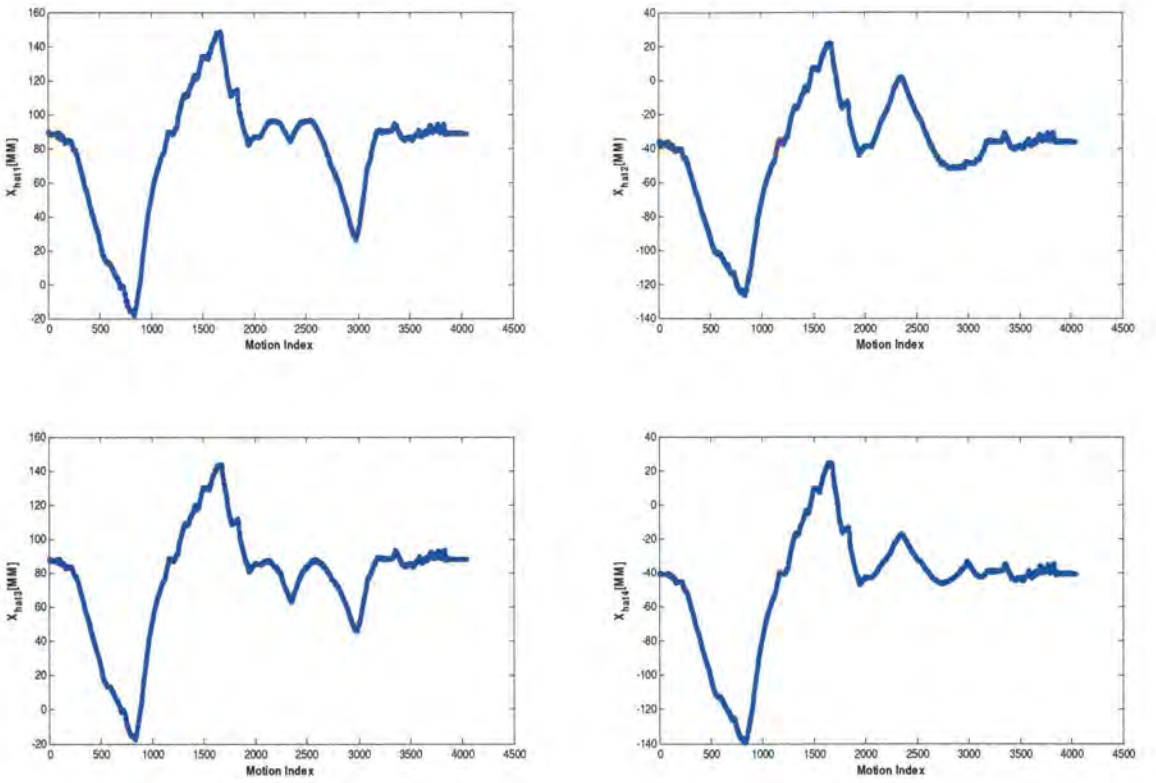


Figure 83 *Translation X for each target point*

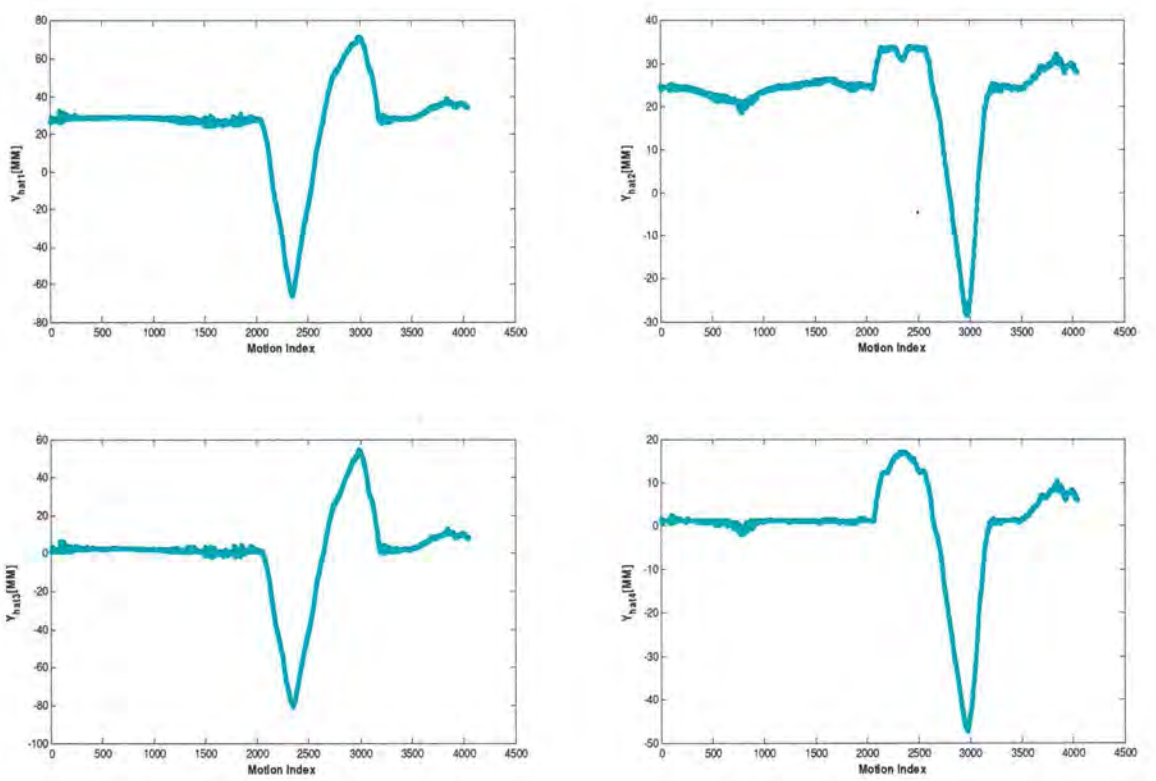


Figure 84 *Translation Y for each target point*

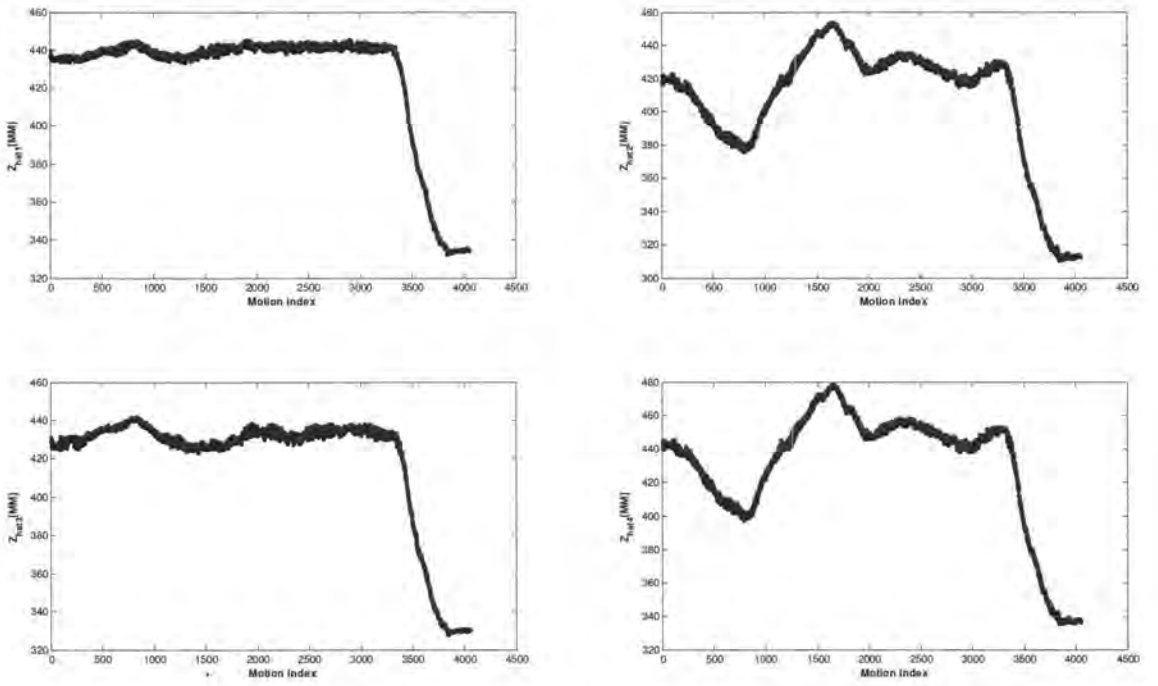


Figure 85 *Translation Z for each target point*

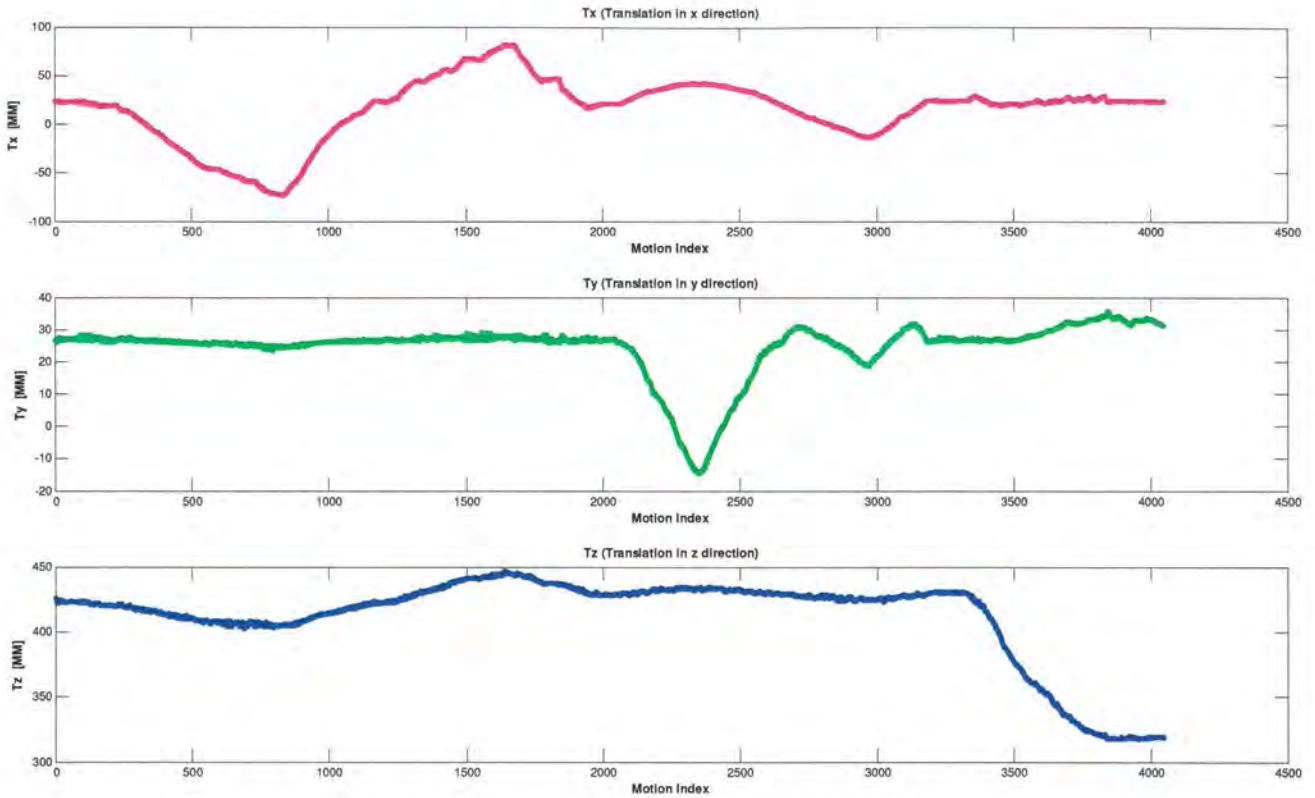


Figure 86 *Translation plots for the entire rigid body*

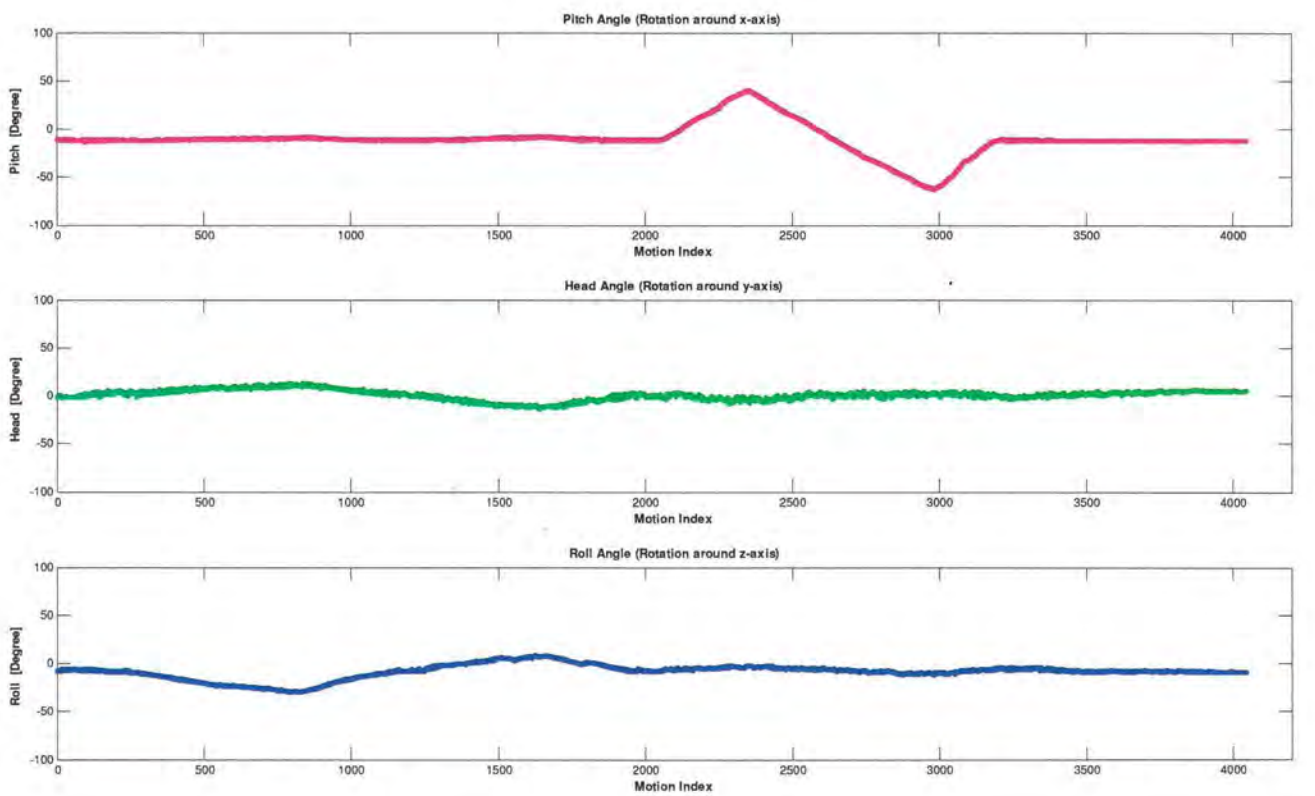


Figure 87 *Rotation plots of the entire rigid body*

5.4 Real time tracking in C

The solution was tested in a real-time 3D audio rendering demo implemented in C. The 3D audio rendering demo has a Graphical User Interface (GUI) that visualizes the tracking performance in 3D space from 3 different viewing positions. The GUI window has the projection of 3D space in three 2D planes and each plane displays the translation and angles in that plane.

The GUI also allows the user to configure the source and scene properties of the virtual 3D audio environment.

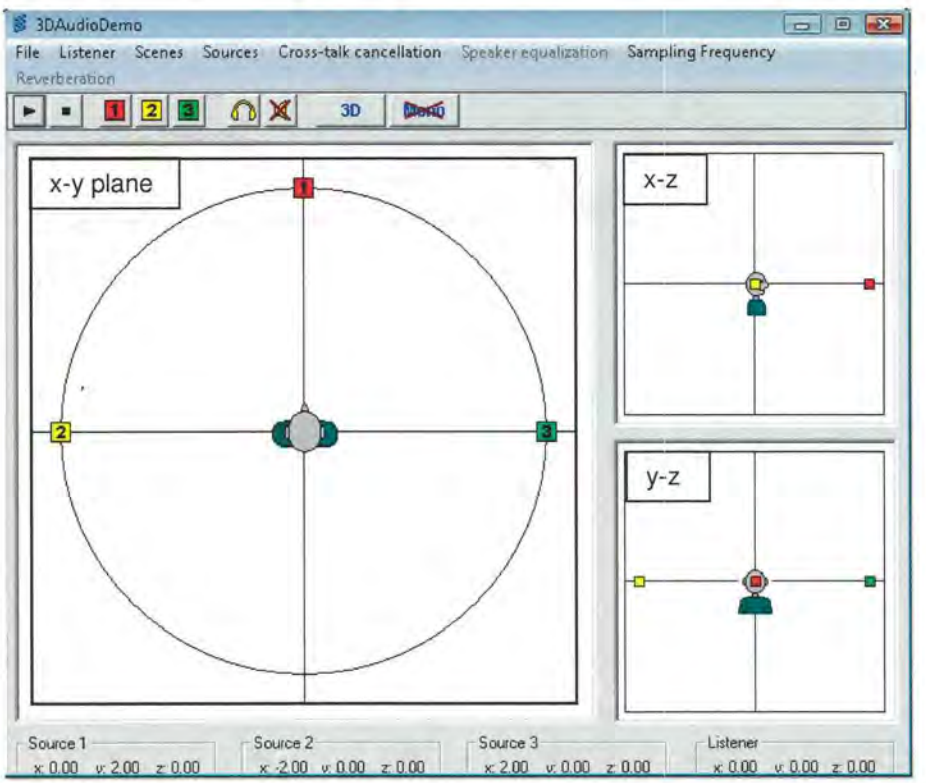


Figure 88 *The virtual 3D audio GUI*

For example, in the source menu, some of the source properties such as sound, path, position, distance attenuation and similar parameters can be modified as shown in figure 89:

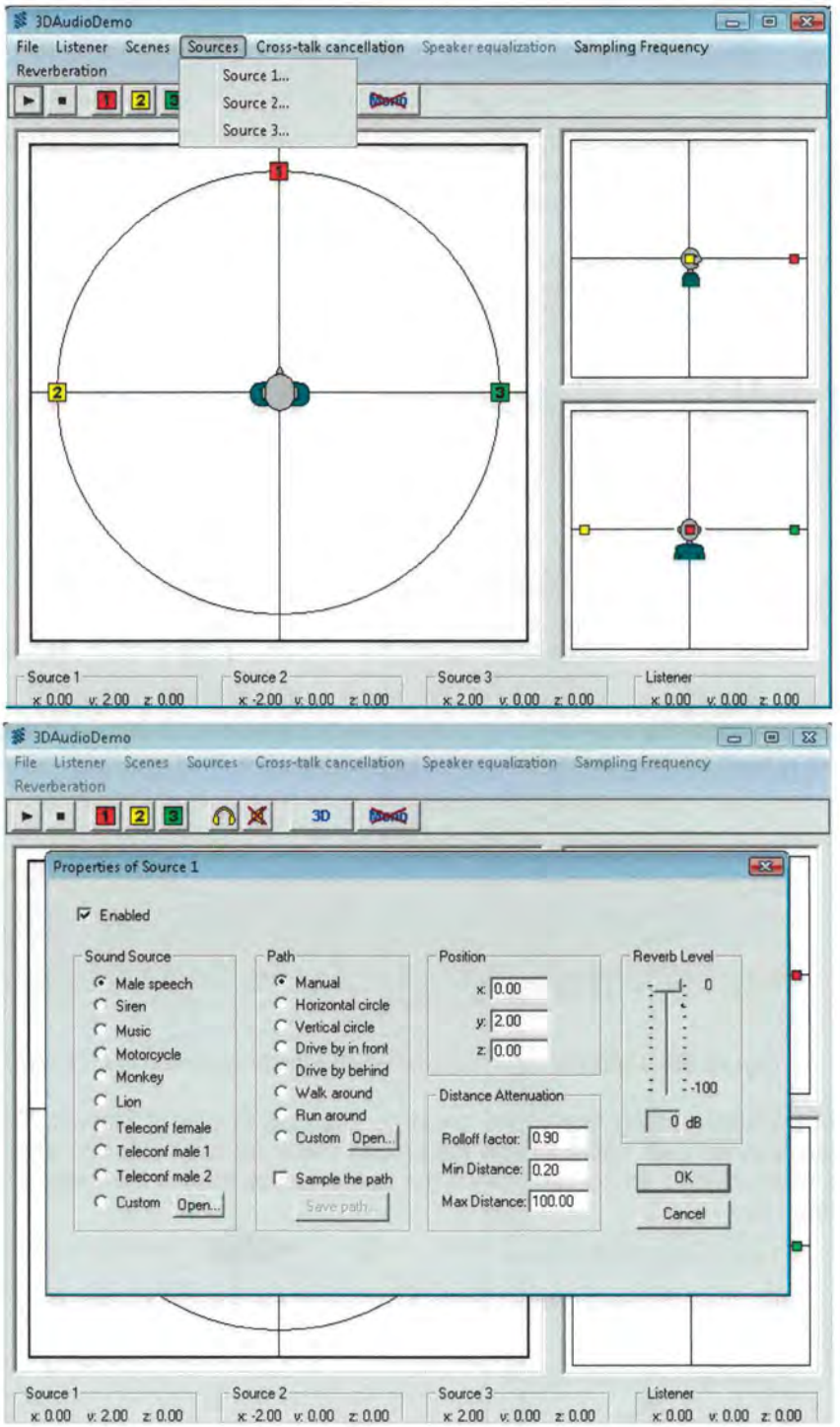


Figure 89 *The GUI provides the possibility of modifying the source properties based on user's interest.*

In the Reverberation menu, different reverberation environments can be chosen as shown in figure 90 below:

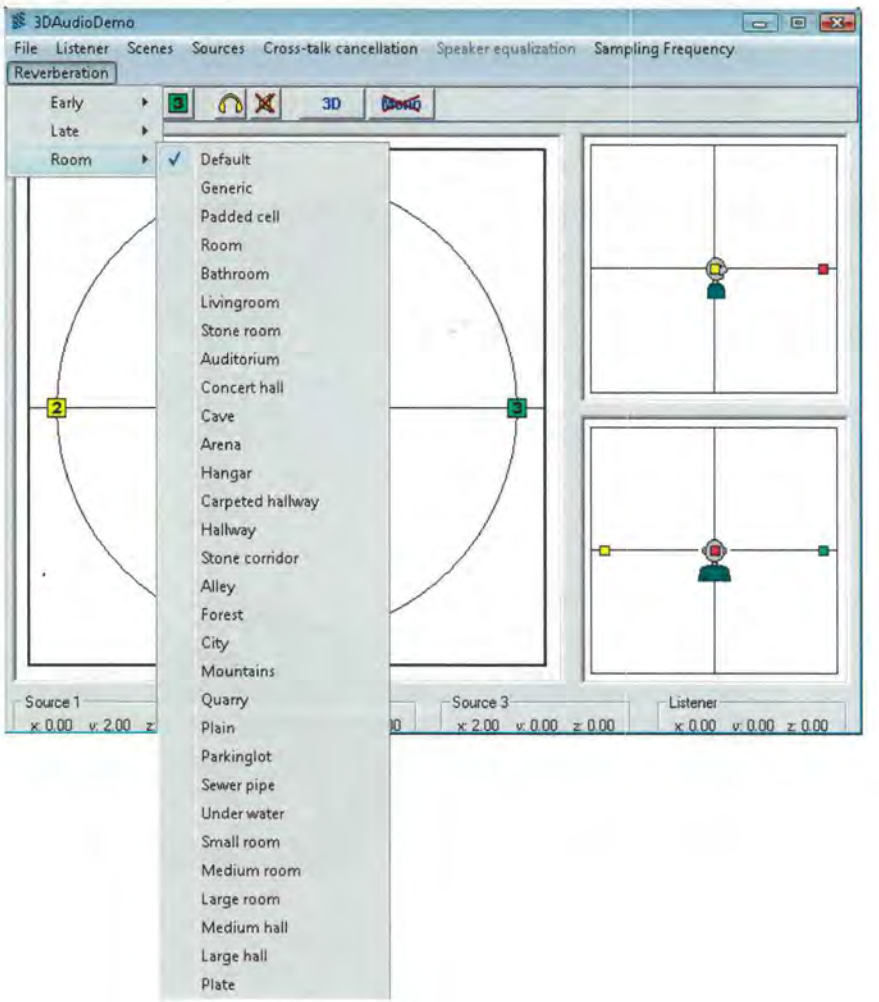


Figure 90 *Different room reverberation effects specified in GUI window*

Similarly, in the scene menu, the scene could be changed to one of Manual, Circles, traffic, Zoo, Teleconference mode (see figure 91). In the Circles mode for example, one of the sources moves around the listener along an x-y circle while the scene moves around the listener in an x-z circle as shown in figure 92.

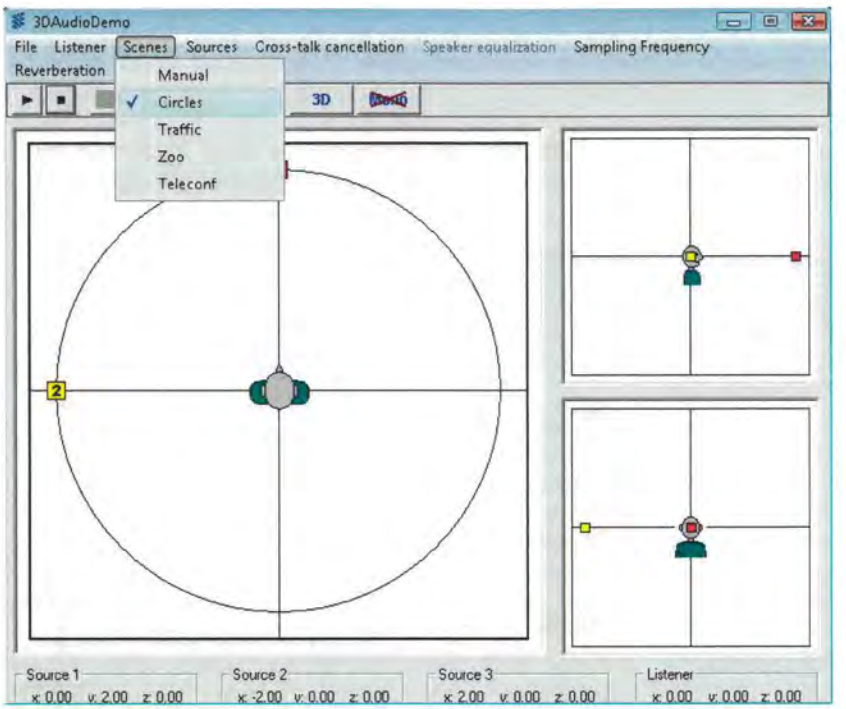


Figure 91 *The virtual 3D audio scene being configured on Circles mode*

In the Circle scene, source no 1 is singing a song while sound 2 is an ambulance making a siren sound. The spatial audio scene rendered for the listener is very convincing and the listener truly "hears" the scene moving around him. Without head tracking, the listener remains fixed in the images.

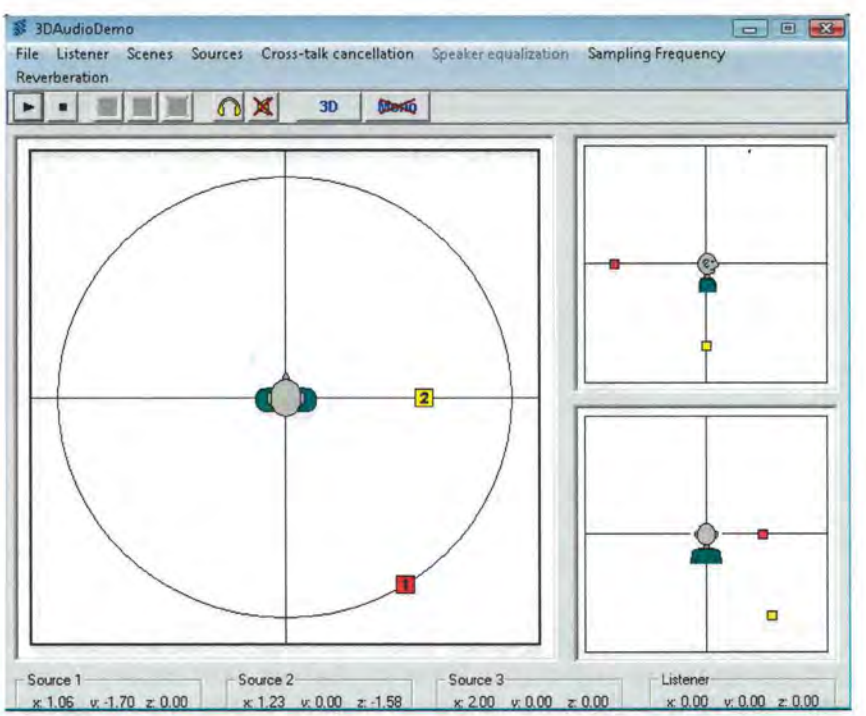


Figure 92 *The virtual 3D audio scene configured in the circle mode. Two sources are moving along x-y and x-z circle around the listener.*

When head tracking is coupled to the 3D audio demo, we are not only able to change the source position but also to move the listener. As mentioned earlier, in a dynamic virtual environment, the 3D audio demo is relative to both listener and source location and thus knowledge of both is necessary.

Figure 93 shows what the tracking results can look like when the head tracking algorithm is active. Each 2D plot illustrates the movement of the listener in its plane. For rotation, each image plane indicates one of the angles and thus for 6DOF visualization, all images should be considered together.

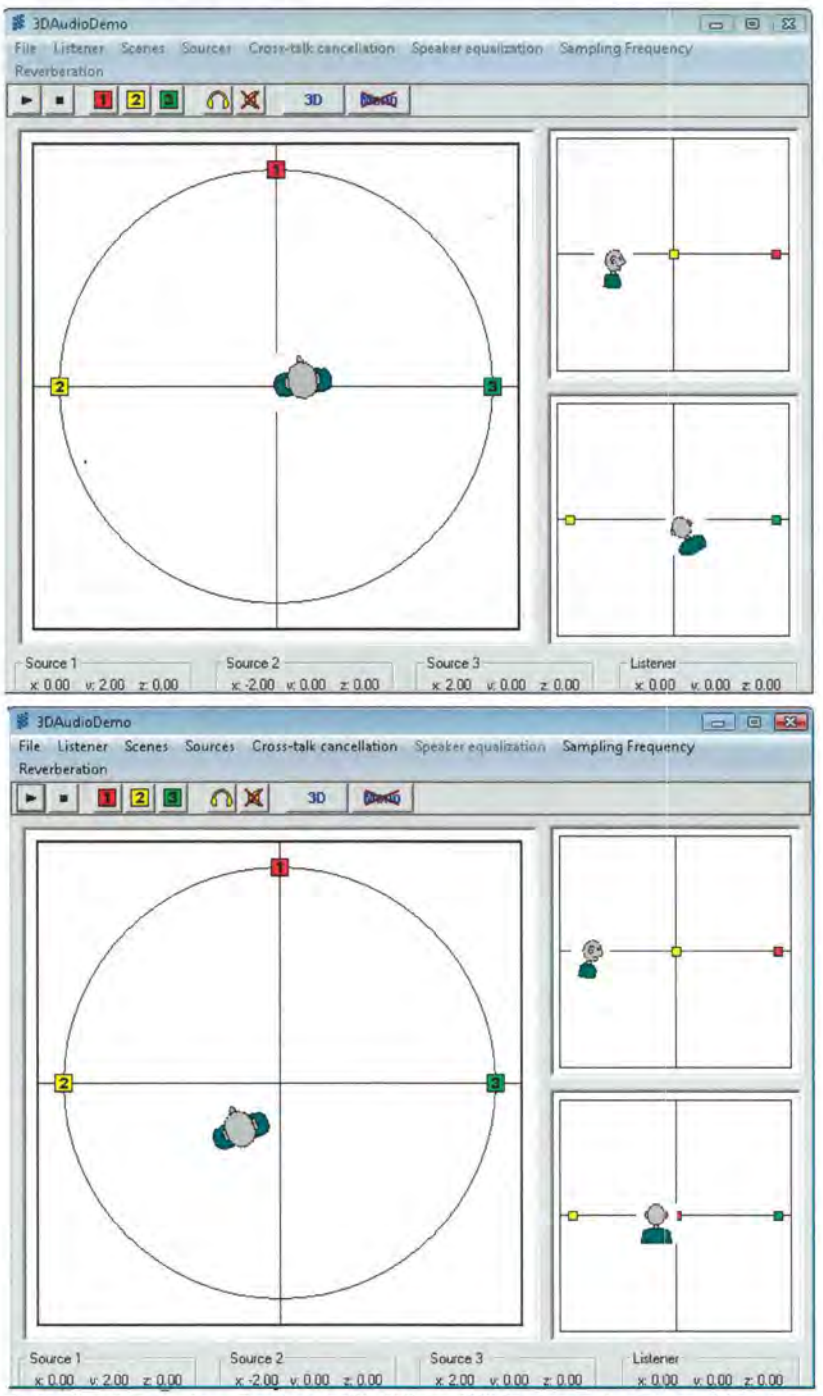


Figure 93 *3D Audio Rendering Engine GUI. When Wii Remote head tracking is active, the user of the 3D audio demo acts as an input to the 3D audio rendering engine and makes the engine create user-dependent 3D audio scenes. Also, synchronized with user's movement, the listener images translate and rotate in different 2D planes in such a way that the movement is in full agreement with listener's position and orientation in 3D space.*

6 Conclusion and Future Work

6.1 Conclusion

A head tracking system was built using the Wii-remote and a wireless headphone with some IR-LEDs mounted. This is an inexpensive solution because it only requires a Wii remote which costs less than 25 Euros and some IR-IEDs which can be bought at a few cents each. The total cost of the complete head tracking system is only about 25-30 Euros which is really inexpensive compared to compatible commercial head tracking systems.

The Wii remote uses a Bluetooth connection. This gives the Wii remote more freedom in its movement without a need to be connected to the computer with a cable or USB connection. The controller is also equipped with a high resolution 1024*768 pixels^2 camera which functions at the rapid rate of 100 HZ. The camera is equipped with an onboard camera-chip in which the chip features an integrated multi-object tracking. The onboard chip makes the IR detection (or bolb-detection) really easy with low computational power. Thus it is very easy to establish a tracking system with IR-sources in contrast to conventional tracking system where IR-detection needs to be implemented.

The key problem in head tracking is the pose estimation problem. The pose problem by definition refers to the process of determining the position and orientation of a 3D object from the 2D projection of the object onto the image plane of the camera and a reference 3D model. The pose estimation problem can be solved in different ways. Two common methods for estimating the 3D pose from the image plane data were presented known as "The Perspective N-Point Problem (PnP)" and "Direct Rigid Body Transformation". These methods have different problem formulation and use their own system of nonlinear equations. We tested and compared two methods on both syntactic and real camera data and the PnP method was selected in our final implementation. An iterative solution to PnP was also presented.

For this project, the head tracking solution was developed as software library in C. The software is the application-independent meaning that it is easy to reuse the head tracking functionalities from other applications.

In order to demonstrate the head tracking system in a real-time application, the library was integrated into Ericsson's 3D audio engine. The engine is equipped with all functionalities needed to simulate the wave propagation in a 3D environment. It simulates the right and left eardrum signals which trigger spatial hearing of the listener. The engine can in fact model the source and listener behaviour in a virtual environment. With a head tracking solution in place, the listener will act as an input to the head tracking algorithm and enables the 3D audio engine to generate user-dependent audio scenes. As a result, when the listener moves in the room, the information about his head position and orientation is sent to the 3D audio engine which in return enables the engine to produce the binaural signals in full agreement with listener's location in the space. When listening to the sounds the listener will experience the sense of actually being in a 3D environment and perceives sounds as coming from different directions around him.

In relation to a closely related work by Johnny Chung Lee on Wii-Remote for head tracking (can be found in YouTube), this work tracks the full 6DOF movement of the listener (translation + orientation) while Chung lee's work only tracked the translation (3DOF) assuming that the orientation was the negated translation.

6.2 Future Work

Future game and VR applications will count on innovative new technologies to revolutionize the way users and applications interact. One such a technology is the Wii remote from Nintendo. The Nintendo Wii remote is a 25 Euro device that contains a high resolution IR camera, the Bluetooth connection and an internal accelerometer. The combination of these features can be configured for a wide range of applications.

Based on key technologies that the Wii remote introduces, it can be exploited for two specific categories of applications. The ones that benefit from the wireless connection and those which lend themselves to motion sensing capability. Head tracking is a new application from the first category that was studied in this project and it was shown that 6DOF tracking is possible just by using pure camera data. In the future, a variety of applications can be developed that make use of the head tracking library. As for the second category, for example, it is thinkable to use the device as a replacement for a mouse for data manipulation and displays.

There head tracking system itself can be improved in several ways. The camera limited Field Of View (FOV) is clearly a problem. One possible way to tackle that is to mount the Wii remote device on special servomotors known as *Pan and Tilt* servos to rotate the device so that the camera target is always in the camera's FOV.

The current solution uses one Wii remote to detect the position of IR-lights. The controller is mounted on a stationary holder and determines the position and orientation of IR-lights. With two cameras *stereovision* techniques can be used to extract 3D information. In this case, *triangulation* determines 3D pose of points using camera parameters. Using two Wii remote controllers and a stereovision algorithm can also be a remedy for the camera's limited FOV and an improvement toward robustification of the tracking algorithm.

8 Appendix A "The Nintendo Wii"

8.1 Introduction

In November 2006, the game company Nintendo released its newest video game console, the **Nintendo Wii**[14]. The company's previous game console GameCube had not fared well in terms of market share in competition with the powerful alternatives released by its competitors, Microsoft and Sony. However, when the Wii console was introduced it was a revolution in the console world such that in less than one year it became the market leader of its console generation, selling over 20 million units worldwide and 100 million units by 2009. The success was largely attributable to the innovative interactive technology introduced by the console controller, the **Wii Remote**.

Figure 94 *The Nintendo Game Consoles: GameCube (Left), Nintendo Wii (Right)*

The Wii Remote introduces amazing game interaction capabilities that make the games fun to play. The motion of the Wii remote is used as the main input for the games and so interacting with a game by moving the Wii Remote around feels much more natural. In Wii tennis, for example, users swing the controller (the Wii Remote) as if it were a tennis racket. These motions are sensed by the sensors in the controller and transmitted wirelessly to the game console which renders the player on the screen as a game character swinging a tennis racket, in the same arc, with the same speed, hopefully hitting over the net. The remote is frequently used as a pointer. Common Wii game interactions using the controller as a pointer include selection, navigation, aiming a weapon or tool, drawing, rotating objects, and push-pull interactions as shown below:

[14] Pronounced /ˈwiː/ like the pronoun we

Figure 95 *Motion-Game Interaction using the Wii Remote*

8.2 The Wii Remote

8.2.1 Why is it significant?

The Nintendo Wii Remote is a handheld device resembling a television controller and represents a new class of sensor device. In addition to buttons, it combines a high-speed high resolution IR camera with a point tracker system-on-a-chip and a 3-axis accelerometer into an inexpensive device that can be trivially and wirelessly connected to most computers, through Bluetooth connectivity [3].

Figure 96 *The Nintendo Wii Remote*

Before its release, acquiring an equivalent system in a *ready-to-use* form may have cost thousands of dollars, but this amazing set of sensing, interaction and communication system technologies makes the Wii Remote one of most sophisticated PC-compatible devices that offers researchers a *simple* motion-based input device today that can be easily exploited for investigating user interaction researches.

The Wii users hold the Wii Remote in one hand and point it to a television screen that has the sensor bar either above or below the screen. However, many Wii applications have reversed the role of the sensor bar and the remote and use the controller in a stationary mode with the IR lights moving. In this way, Wii technology has been used to develop for example, head-tracking tools which monitors the position of LEDs mounted on user's head and creates a highly realistic, 3D virtual environment based on feedbacks from the controller.

Overall, the main significances of the Wii lies in the potential it offers for creating new devices and applications based on the location- and movement- sensing ideas. The technology provides an interactive research experience for student and researchers to practice and learn in a *low-cost, low risk* manner.

8.2.2 What is the role of sensor bar

In its typical application, the Wii Remote uses a so called 'sensor bar' to get the IR light information in order to sense where it is pointing to at the screen. However, the term 'sensor bar' is somewhat a misnomer because the device (sensor bar) does NOT contain any sensors and it does NOT sense anything; rather in contains two groups of infrared LEDs at each side. The Wii remote camera sees two groups of dots and provides the relevant information about the dots (x,y) coordinates in the image plane of the camera.

Figure 97 *The Wii sensor bar (Left); Two groups of IR lights seen by the camera (Right)*

Some innovative users use other source of infrared light like a pair of candles or flashlights instead of sensor bar which can also do the job. Other interfering sources of infrared light, however, such as the sun light[15] or incandescent light bulbs can cause detection problems when they are around. Instead, the fluorescent light which emits little or no infrared light can be used to alleviate this problem.

The IR LEDs cannot be seen by the human eye since infrared radiation has a wavelength of (750nm–100μm) which is larger than that of visible light (380nm–750 nm). Some cameras and other devices with a wider visible spectrum can be used to see the LEDs.

8.2.3 Main features of the Wii remote - functionality

The Wii Remote's official specifications are unpublished and Nintendo has released few technical details about the Wii system. However, the global reverse engineering community has collectively obtained a significant portion of the technical information through reverse-engineering. Much of this information is collected in online wiki http://wiibrew.org.

Much of the information provided in the following description about each Wii remote component represent only higher-level information that are related to building custom applications.

[15] The sun light contains 47% share of the spectrum for infrared light.

8.2.4 Infrared camera –IR tracking

In the tip of the Wii Remote, behind the dark window lies an Infrared camera sensor that is used to track infrared light sources. This section summarizes the main specifications of the camera:

1. **The processor and the camera resolution**

The Wii remote camera chip is manufactured by PixArt Imaging. The chip contains a monochrome camera with built-in processor which features an integrated multi object tracking (MOT) engine which provides high-speed tracking of up to **four** simultaneous IR light sources. The data location of the camera appears to have a resolution of 1024x768 pixels but this seems to be the interpolated resolution. In other words, the built-in processor uses 8x subpixel analysis to provide 1024x768 pixels resolution for the tracked points and so the genuine resolution is only 128x96 pixels.

Figure 98 *The PixArt IR Camera chip. Integrated Multi-object tracking minimizes wireless data transmission*

2. **Camera frame rate**

The camera operates at a high frame rate of 100 Hz which outperforms similarly priced webcams tracking at 30 Hz. However, this high-frame rate generates a lot of data which needs to be analysed. Assuming that one byte of memory places one pixel in itself, sending 100 images of resolution 1024x768 pixels takes approximately 630 Mbit/sec.

100(images) x1024 x 768 x 8(bit) = 629.1 x 10^6 bits
Bit rate = ≈630 M bit/sec

This rate is extremely high to be transmitted via Bluetooth to the Wii. The maximum bit rate that is possible to transmit over Bluetooth connection is 1 Mbit/sec. The Wii can support up to *four* Wii remotes using the same channel which intensifies the bandwidth limitation even more. So the question is how the Wii Remote is able overcome this difficulty?

The Wii remote analyzes the images itself and ONLY sends the *position* and the *size* of infrared light sources detected by the camera to locate the Sensor Bar's point of light. The distance between two clusters of dots in the sensor bar is a fixed distance. Knowledge of this distance along with the information received by the Wii allows the Wii CPU to calculate the relative distance between the Wii Remote and the sensor bar using triangulation. Sending the dots coordinates instead of the image itself saves a lot of bandwidth.

3. Intrinsic[16] camera parameters

These parameters are the internal characteristics of the camera that are constant during tracking. Detailed explanations of camera parameters can be found at section 2.2.1. As mentioned before, the cameras intrinsic parameters are not published officially but some of them like focal length and centre of projection are crucial in expressing the camera's projection equations[17] (section) used for tracking.

In practice, it is very difficult to calculate these parameters. However, while looking at the camera's projection equations, these values appear as ratios of one another. This means one can express the focal length and centre of projections in units of pixel instead of some physical units.

In this respect we made some assumptions that seem to work well in reality. We assumed the pixels in x and y direction have 1:1 ratio, meaning square pixels (p_x=p_y). The other assumption was to think of centre of projection on the image plane reside at the center pixel (512,384).

In addition, the IR camera has a limited horizontal Field Of View (FOV) approximately about 45 degrees.

4. Band pass filter and blob detection

The IR camera detects only radiations with wavelength in the range of infrared light. This is achieved by using a *band pass* filter in front of the camera that only lets infra red light through. Using a filter, sources with 940nm wavelength are detected with **twice** the intensity of 850nm sources. If IR-pass filter is removed, it will detect any bright objects. By trial and error we know that the filter has a peak frequency of 940nm with a bandwidth of about 40nm.

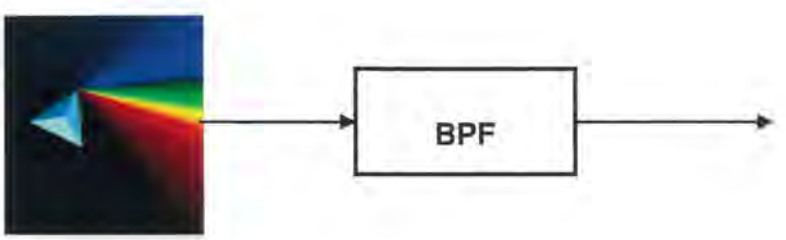


Figure 99 *IR-Camera Band Pass Filter*

On the other hand, the IR camera uses a *blob detection* technique to extract images from the camera's image plane. Blob detection is a technique used in *computer vision* to detect points or regions in the image with different intensities compared to the surroundings, e.g. brighter or darker. A blob is a connected region in computer vision.

Blob detection is done fairly easily in the IR camera by using IR-Pass filter, since no noise exists on the images and the perceived image is fairly clean.

[16] Sometimes called **Internal** parameters
[17] Pin Hole Camera model

8.2.5 Accelerometer– motion sensing

The Wii remote is equipped with a 3-axis linear accelerometer[18] that provides the motion-sensing capability of the device. When the Wii remote is at rest horizontally, it reports the acceleration due to pull of gravity[19] but in opposite direction (+Z).

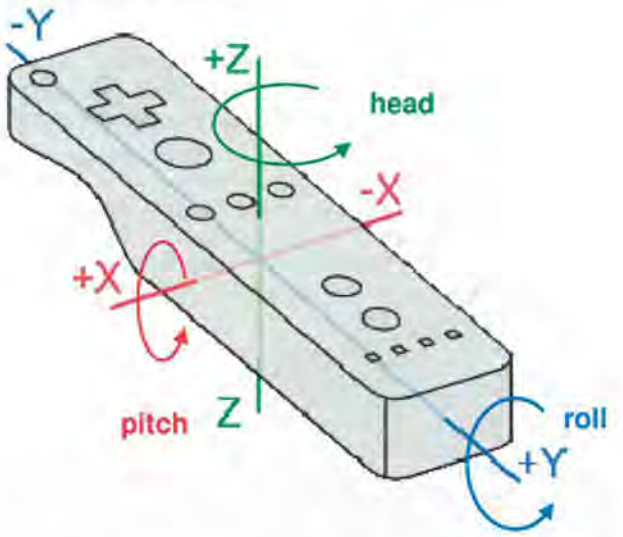


Figure 100 *Coordinate system used by the Wii Remote*[20]

This fact could be used to measure tilt from the acceleration. Among the orientation angles, pitch and roll can be easily detected but the third angle *head* remains unknown because the gravity force remains unchanged while rotating around the z-axis. However, the Wii remote uses an additional sensor in order to get information about that angle and thus the Wii remote can detect acceleration from all three axes.

The accelerometer has a +/- 3 g with 10% sensitivity range. At a refresh rate of 100 Hz, it uses 8 bits per axis to report the data.

8.2.6 Bluetooth- wireless communication

Unlike most game controllers, the Wii remote connects wirelessly to most computers without special hardware. This connection is realized with Bluetooth, a well-established wireless protocol for exchanging data over short distances.

Bluetooth uses the (ISM[21]) 2.4 GHz short-range radio frequency bandwidth[22]. Compared to other wireless protocols like Wi-Fi, Bluetooth has a lower bandwidth but it has lower power requirements as well.

The main application for Bluetooth is between a Human Interface Device (HID)[23] and a computer. A Bluetooth-HID standard[24] is used to enable communication between the HID device and the

[18] Accelerometer ADXL330
[19] Approximately 9.81 m/s^2
[20] Source : http://wiibrew.org/wiki/File:Wiimote_axis2.png
[21] Industrial, Scientific and Medical
[22] Source : http://en.wikipedia.org/wiki/Bluetooth
[23] Computer devices that interact directly with humans such as keyboards, mice etc.
[24] A modified version over USB-HID

computer. The Wii remote however, uses a Broadcom chip 2042 in order to conform to the Bluetooth HID standard.

Using the standard HID protocol, it is really easy to establish the connection between the Wii remote and a regular personal computer. All that is needed is a Bluetooth dongle and then the connection to the controller is fairly easy. The Wii remote should be placed in discoverable mode by pressing the 1 and 2 buttons at the same time. Once in this mode a number of LEDs based on the battery level will blink, figure below:

Figure 101 *The Wii remote in discovery mode*

Once in discovery mode, the Wii remote can be queried by the host computer within 20 seconds. If no connection happens during that time, the Wii remote will turn itself off. It can be turned on again by pressing the buttons 1 and 2. Continuous pressing of these two buttons will force the Wii remote not to turn off and stay in discovery mode. This is the basic requirement for synchronization of a Wii remote the rest is standard Bluetooth pairing, figure 102:

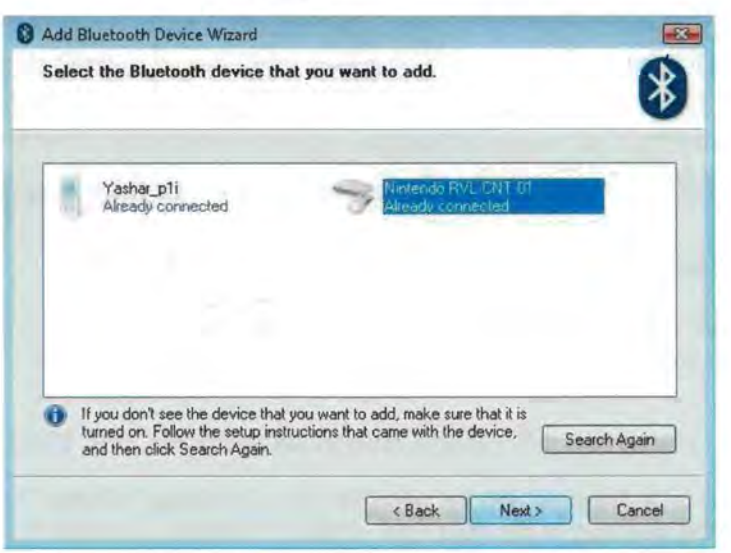


Figure 102 *Synchronizing the Wii remote with computer*

Once a pairing is synchronized successfully, the configuration is quite reliable. This easy way of connection has facilitated the development of many custom applications. For example, an application

has been developed by developer communities that enable users to use a Wii Remote as a mouse pointer on their screen from their personal computers as depicted in the figure below:

Figure 103 *Custom applications for the Wii remote as mouse Cursor Control (Left), Flash-Based Mouse Games (Right)*

8.2.7 The Wii HID interface

Nowadays, the protocol that the Wii remote uses to communicate with the Wii console is known to everybody within the Wii community. The Wii remote uses the standard Bluetooth HID protocol which is directly based on USB-HID. This section provides a general overview of the HID protocol and how the devices with Bluetooth communication use this protocol to communicate with the host. It also explains the HID reports that are sent from the Wii remote to a Bluetooth host.

In some non-volatile segments of an HID device memory, the information about the device is stored. These segments are called HID *Descriptor* Block and they allow devices to be self-describing. The descriptor is used to allow devices to be identified as belonging to one of a finite number of classes.

An HID descriptor identifies the number, type and size of the *Physical descriptor* and *Report* that is associated with a HID Class device. The host program uses a corresponding HID class driver to retrieve and route all data as shown in figure 104. This is accomplished by examining the descriptor of the device and the data it provides.

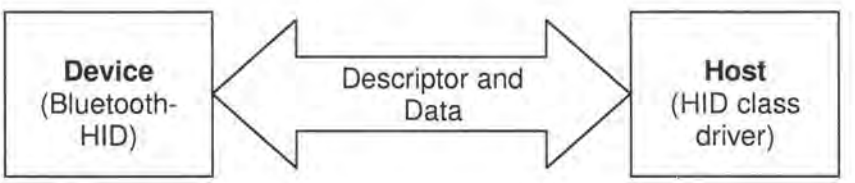


Figure 104 *A HID class device uses a corresponding HID class driver to retrieve all data [36]*

- **Physical descriptor**

A *physical descriptor* is an **optional** descriptor that identifies the part of the human body that activates the controls on the device [25].

- **Report**

A *report* is similar to a network port responsible for a particular service. However, reports are *unidirectional* and the direction of the data flow - **Input** or **Output** - is listed in the HID descriptor. A report descriptor describes each piece of data that the device generates as well as the data that is actually being measured.
The HID class driver examines an item's report descriptor to determine the size and composition of data reports coming from the HID class device.

[25] HID control

8.2.8 The Wii remote reports

When the Wii remote is queried by the host program[26], the Wii remote reports a great deal of information. It particularly reports:

Name	Nintendo RVL-CNT-01
Vendor ID	0x057e
Product ID	0x0306

***Table 1**: The Wii remote reports the above information when queried by the host*

The maximum number of reports is 100 per second and no security or authentication of standard Bluetooth HID is required.

Each report is specified by a header – **Report ID** – and the actual data[27] that is being transmitted. The latter is referred to as **Payload**[28] and its size and content is determined by the report ID as shown in

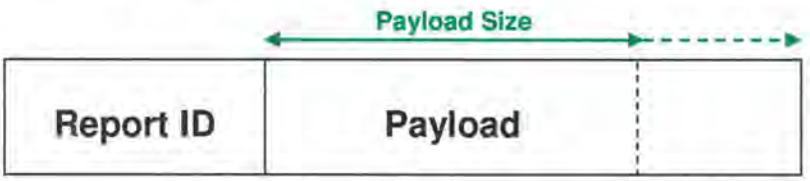


***Figure 105** Data packets structure in the Wii remote rreports. The size of Payload is dependent on Report ID type.*

For example, when the button *plus* (report ID: 0x30) is pressed the following HID input data packet is received in which 00 08 is the **two-byte** payload associated with report ID (0x30):

30	**00**	**08**

Similar to all HID devices, the Wii remote reports its HID descriptor block when queried. The HID descriptor identifies the direction of data blocks for each port as well as the payload size for each port. If the host *receives* the report packets from the Wii remote, it is an **Input Report** whereas if the host *sends* the report to the Wii remote it is an **Output Report**:

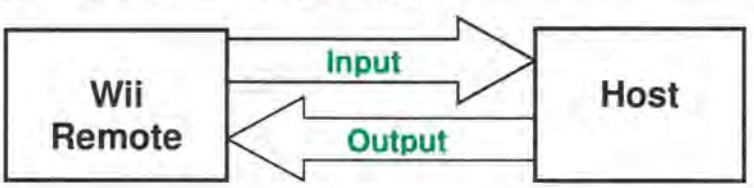


***Figure 106** Report types (Input/Output) - the Wii remote*

[26] With Bluetooth Service Discovery Protocol (SDP)
[27] Sometimes called the *Cargo*
[28] Also called Channel ID

1. Output reports

The following table summarizes all the output reports that the Wii remote uses when communicating to the host. These are data packets sent to the Wii remote from the host. The corresponding payload size is given for each report.

Report ID	Payload Size (Byte)	Function
0x11	1	Player LEDs
0x12	1	Data Reporting Mode
0x13	2	Enable IR Camera
0x14	1	Enable Speaker
0x15	1	Controller Status Request
0x16	21	Write data in Memory
0x17	6	Read data from Memory
0x18	21	Speaker Data
0x19	1	Mute Speaker
0x1a	1	Enable IR Camera 2

Table 2 *The Wii remote Output Data Reports*

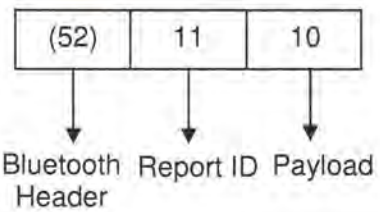
We can consider an example of how the reports IDs are used. For clarity, our convention is to show data packets written in hexadecimal separated by spaces according to the following structure:

(Bluetooth Header)	Report ID	Payload

Example:

There are four blue LEDs on the front face of the Wii remote. These LEDs which can blink at a fixed rate exist to provide special functionalities, such as showing the battery charge remaining. However, the LEDs can be independently controlled by the host and display any patterns.

The following will Turn ON the first LED ⟶ ■ □ □ □



This is a DATA output packet (0x52) on channel 0x11 with the one payload 0x10. The bytes in parentheses are not used when using higher level HID functions rather than Bluetooth functions.

The four LEDs are controlled by the last four bits of the payload. Bit 4 controls the first LED, and bit 7 controls the last one:

Bit	Mask	LEDs
4	0x10	■ □ □ □
5	0x20	□ ■ □ □
6	0x40	□ □ ■ □
7	0x80	□ □ □ ■

***Table 3** : Bit assignment for LEDs*

- **Common feature of output reports:**

Output Reports share a common feature. The third bit of the first byte is an ON/OFF flag for a specific feature such as: Data Reporting Mode (0x12), IR Camera Enable (0x13), Speaker Enable (0x14), Speaker Mute (0x19), IR Enable 2 (0x1a).

The OFF mode is activated by sending 0x00 and the ON mode by sending 0x04. For example:

Function	Hex code
Enable IR Camera	(52) 13 04
Disable IR Camera	(52) 13 00
Enable the speaker	(52) 14 04
Disable the speaker	(52) 14 00

***Table 4:** Some functions and associated Hex codes*

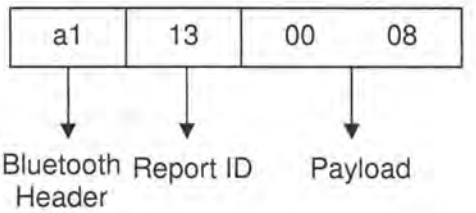
2. Input Reports

The following table summarizes all the output reports that the Wii remote uses when communicating to the host. These are data packets sent to the Wii remote from the host. The corresponding payload size is given for each report.

Report ID	Payload Size (Byte)	Function
0x20	6	Expansion port
0x21	21	Read Data
0x22	4	Write Data
0x30-0x3f	2	Data Reports

Table 5: *The Wii remote Input Data Reports*

For example when the **Plus** button is pressed, this HID Data packet is received:



And when it is released, this packet is received:

a1	13	00	00

The bit assignment for the buttons is shown in table 6:

Button	Number (dec)	Value (hex)
②	1	0x0001
①	2	0x0002
B	3	0x0004
A	4	0x0008
⊖	5	0x0010
Home	8	0x0080
←	9	0x0100
→	10	0x0200
↓	11	0x0400
↑	12	0x0800
⊕	13	0x1000

Table 6: *Bit assignment for different buttons on the Wii remote*

8.3 Software libraries - development of custom applications

In the light of the Wii remote's compatibility with the Bluetooth HID specification and discoveries made on the Wii remote communication protocols, many custom applications have been developed for the remote. On the other hand, the open development community has created several software libraries that can connect to the Wii remote and control the remote. These libraries are available on nearly every major development platform such as Windows, Mac OS, and Linux. The libraries the programmer with much work when creating a custom Wii application.

The Bluetooth HID driver interfaces the Wii remote with the computer using a custom report format as illustrated below:

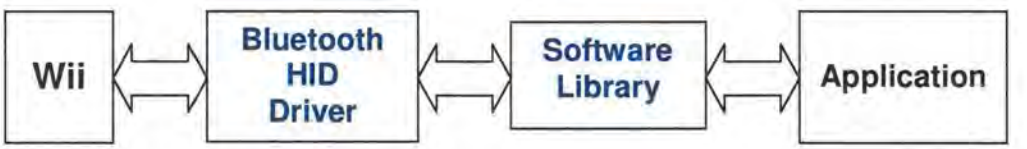


Figure 107 *Development of a custom application for the Wii remote*

One of the open-source libraries for using the Wii remote functionalities is **Wiiuse**. Wiiuse is written in C and is a cross platform library (Linux and Windows). It is single-threaded program and has a clean API. It uses non-blocking and thus the calls do not block during the application runs, a case which can cause the whole application to halt.

These libraries are in active growth and since the software APIs might change rapidly, we will not going discuss them in details. The reader is referred to (Wii homebrew) for the updated information. Most of the libraries are available for free download.

The applications however, use a number of features in the remote to allow development of the Wii-remote based applications. A graduate PHD student at Carnegie Mellon University, Johnny Lee, has pursued interesting research studies on exploring novel technologies that enhance the practicality of interactive technology. He has developed a number of Wii-remote based applications which seems to be unique in their own types. Some examples of his Wii remote-based applications are listed below and shown in the following figures[29]:

- *Interactive whiteboards and tablet displays*
- *Head tracking for VR displays*
- *Spatial augmented reality*

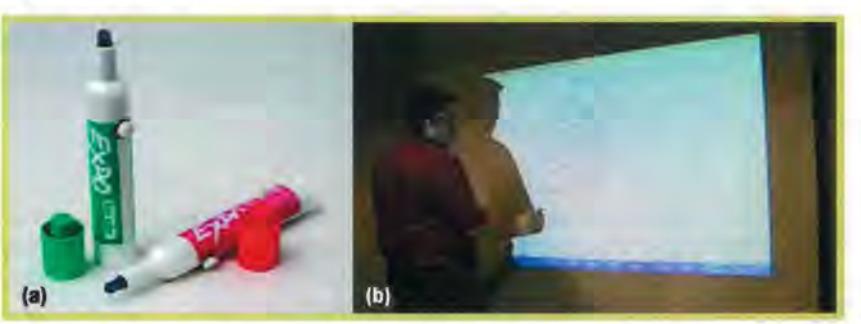


Figure 108 *Interactive display (a) Infrared LED pens used for (b) interactive whiteboard*

[29] Source: Hacking the Nintendo Wii remote (Lee, 2008)

Figure 109 *Desktop VR. (a) Rigid IR emitters on glasses (b) rendered scene by the Wii that depends on the viewing angle*

8.4 Head Tracking with the Wii Remote

An accurate estimation of head position and orientation (pose) in 3-Dimensions is in important task in many applications and is known as *Head tracking.* Head tracking is a special area of motion tracking in which only a small portion of the body (the head) is tracked. Since a head does not have any joint it can be considered as rigid body which has only one position. This can significantly simplify the system since we can concentrate only on one point.

One of the applications of head tracking is in 3D Virtual Reality[30] environments. Virtual reality technology allows the user to interact with an environment that is simulated by a computer and models the real world. The virtual reality environments are usually **Visual** and displayed on computer screen or stereoscopic displays, or **Auditory**[31] and produced through loudspeakers or headphones. A well known audio clip of Virtual Reality Audio is provided below called Virtual Barbershop. It is a very interesting sound file which gives you the illusion of being in a real Barbershop by using 3D Audio techniques. Make sure to use headphone while listening to the clip:

ViRTuaL-BaRBeRSHo
P.mp3

Audio File 1 : Virtual Barbershop (Double Click to play)

For a real-time virtual reality system that renders the scenes based on the information about the user location, an important task is to obtain the viewing or listening position of the observer/listener in 3D space. Head tracking deals directly with this problem and can significantly increase the performance of the applications. In such systems, a 3D world is simulated on 2D screens or over auditory devices.

A conventional head tracking system consists of an optical camera and some markers that are attached to the user head. The user can wear a helmet or glasses with optical markers attached. The camera sees the markers and finds them as black spots on its image plane with a white background. The blob detection techniques are used then to recognize if the found spots are ellipsoid. If this is the case, the spots are added to the found list. The information about the markers position helps to estimate the head position (and orientation) in the 3D space.

One example of such a system is the Wii remote optical camera technology that can be utilized for head tracking purpose. If we place the Wii remote at a fixed place (see modes of tracking) and use a set of wearable IR emitters on the user's head, we can apply computer vision techniques to obtain 3D tracking data from individual bright spots. This means that we can track the head location relative to the controller and reconstruct the head pose in the 3D space.

[30] VR
[31] Called Virtual Reality Audio System

8.5 Modes of Tracking with the Wii Remote

In its typical application, the Wii users use the Wii remote in one hand and point to a television screen that has the senor bar near the screen. In this mode, the controller's internal accelerometer is used to obtain the controller's relative orientation with respect to the sensor bar.

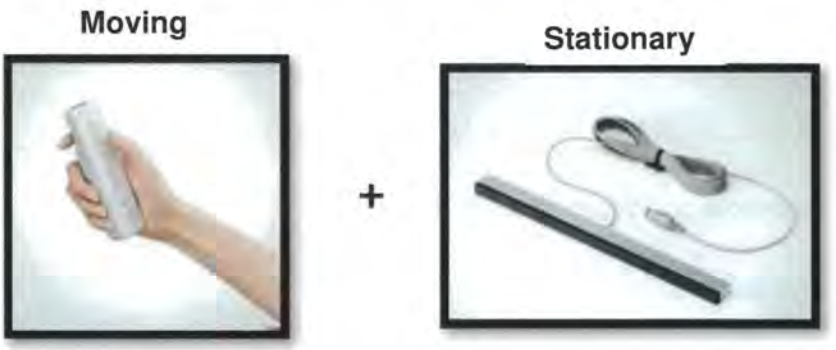


Figure 110 *Conventional mode of tracking using the Wii remote*

However, many Wii users have reversed the ordinary mode of tracking and instead use the Wii remote in stationary status and allow the IR emitters move in the 3D space. This mode is a suitable candidate for object tracking purposes since the only thing required in this configuration is to mount some IR emitters on the object that should be tracked.

For this reason, many motion-capture tracking systems use this mode of tracking which as a result transforms the them into high-performance motion tracking system. Our head tracking solution is also based on this mode of tracking.

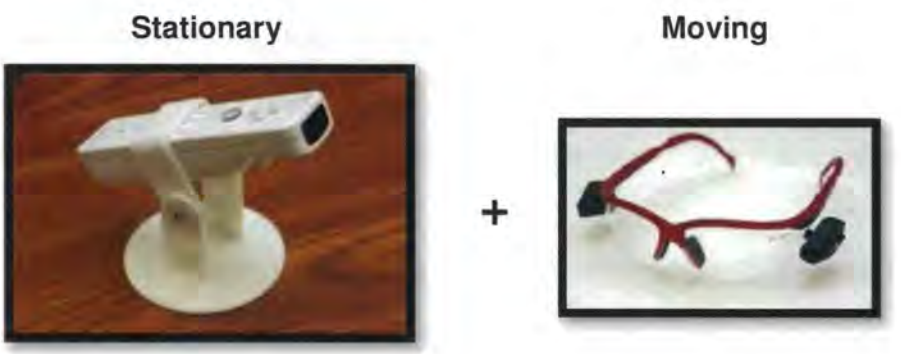


Figure 111 *Modes of tracking using the Wii remote*

9 Appendix B "Overview of the Optimization Algorithms"

9.1 Scalar functions and their derivatives

A scalar function is a real multivariable function that takes one or more variables but returns a single value and is denoted

$$F(\mathbf{x}) = F(x_1, x_2, ..., x_n) \qquad F : R^n \to R \tag{58}$$

For example $F(x_1, x_2, x_3) = x_1^{\,2} + x_2^{\,3} + 4x_1x_3$ is a scalar function with three variables.

If $F(\mathbf{x})$ is continuous on an open set $\Omega \subset R^n$ we write:

$$F(\mathbf{x}) \in C \tag{59}$$

If all first order partial derivatives are continuous (on Ω)

$$F(\mathbf{x}) \in C^1 \tag{60}$$

Similarly if all *k*:th partial derivatives are continuous we can write :

$$F(\mathbf{x}) \in C^k \tag{61}$$

9.1.1 The Gradient of a scalar function

The gradient of a scalar function $F(\mathbf{x}) = F(x_1, x_2, ..., x_n)$, $F : R^n \to R$

if $F(\mathbf{x}) \in C^1$ is defined as:

$$[\nabla_x F(\mathbf{x})]_{n\times 1} = \frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} = [\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \cdots, \frac{\partial F}{\partial x_n}]^T \tag{62}$$

The gradient is a vector with the direction towards maximum increase of $F(\mathbf{x})$ with respect to vector $\mathbf{x}$ [32].

9.1.2 Second order derivative of a scalar function

If $F(\mathbf{x}) \in C^2$, $F : R^n \to R$ we can define the second order derivative of a scalar function as below:

$$H(\mathbf{x}) = [\nabla^2_{xx} F(\mathbf{x})]_{n\times n} = \left[\frac{\partial^2 F(\mathbf{x})}{\partial x_i \partial x_j}\right]_{n\times n} = \begin{bmatrix} \frac{\partial^2 F(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 F(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 F_2(\mathbf{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 F_2(\mathbf{x})}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 F(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F(\mathbf{x})}{\partial x_n^2} \end{bmatrix} \tag{63}$$

The second order derivative is known as the **Hessian Matrix**.

[32] All vectors are defined as column vectors.

9.1.3 Taylor series of a scalar function

The Taylor expansion for a multivariate scalar function $F(\mathbf{x}), F_i(\mathbf{x}) \in C^2$ around $\mathbf{x}_0$ is

$$\underbrace{F(\mathbf{x})}_{[\ ]_{1\times 1}} = \underbrace{F(\mathbf{x}_0)}_{[\ ]_{1\times 1}} + \underbrace{(\mathbf{x}-\mathbf{x}_0)^T}_{[\ ]_{1\times n}} \underbrace{\nabla_x F(\mathbf{x}_0)}_{[\ ]_{n\times 1}} + \frac{1}{2}\underbrace{(\mathbf{x}-\mathbf{x}_0)^T}_{[\ ]_{1\times n}} \underbrace{\nabla_{xx}^2 F(\mathbf{x}_0)}_{[\ ]_{n\times n}} \underbrace{(\mathbf{x}-\mathbf{x}_0)}_{[\ ]_{n\times 1}} + h.o.t \tag{64}$$

Note: h.o.t denotes higher order terms.

By skipping higher order terms, we can approximate the derivative

$$\frac{\partial F(\mathbf{x})}{\partial \mathbf{x}^T} = \underbrace{\nabla_x F(\mathbf{x}_0)}_{\text{Gradient}} + \underbrace{\nabla_{xx}^2 F(\mathbf{x}_0)}_{\text{Hessian}}(\mathbf{x}-\mathbf{x}_0) \tag{65}$$

9.2 Vector function and their derivatives

By comparison, a vector function is the one whose range is *m*-dimensional, that is a vector of *m* scalar functions which is denoted

$$\mathbf{F}(\mathbf{x}) = [F_1(\mathbf{x}), F_2(\mathbf{x}), \ldots, F_m(\mathbf{x})]^T \qquad F: R^n \to R^m \tag{66}$$

And $\mathbf{F}(\mathbf{x}) \in C^k$ if $F_i(\mathbf{x}) \in C^k$.

9.2.1 The Jacobian matrix of a vector function:

The Jacobian Matrix is defined for vector functions if $F(\mathbf{x}) \in C^1, F: R^n \to R^m$. Explicitly given *m* set of equations in *n* variables as

$$\begin{cases} F_1(\mathbf{x}) = F_1(x_1, x_2, \cdots, x_n) \\ F_2(\mathbf{x}) = F_2(x_1, x_2, \cdots, x_n) \\ \qquad\vdots \\ F_m(\mathbf{x}) = F_m(x_1, x_2, \cdots, x_n) \end{cases} \tag{67}$$

the Jacobian matrix is

$$\boldsymbol{J}_{m\times n}(\mathbf{x}) = \left(\nabla_{\mathbf{x}}.\mathbf{F}^{\mathrm{T}}(\mathbf{x})\right)^{\mathrm{T}} = \left(\begin{bmatrix} \dfrac{\partial}{\partial x_1} \\ \dfrac{\partial}{\partial x_2} \\ \vdots \\ \dfrac{\partial}{\partial x_n} \end{bmatrix}_{n\times 1} \cdot \begin{bmatrix} F_1(\mathbf{x}) & F_2(\mathbf{x}) & \cdots & F_m(\mathbf{x}) \end{bmatrix}_{1\times m} \right)^{\mathrm{T}}$$

$$= \begin{bmatrix} \dfrac{\partial F_1(\mathbf{x})}{\partial x_1} & \dfrac{\partial F_2(\mathbf{x})}{\partial x_1} & \cdots & \dfrac{\partial F_m(\mathbf{x})}{\partial x_1} \\ \dfrac{\partial F_1(\mathbf{x})}{\partial x_2} & \dfrac{\partial F_2(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial F_2(\mathbf{x})}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial F_1(\mathbf{x})}{\partial x_n} & \dfrac{\partial F_2(\mathbf{x})}{\partial x_n} & \cdots & \dfrac{\partial F_m(\mathbf{x})}{\partial x_n} \end{bmatrix}^{T}_{n\times m} \qquad (68)$$

$$= \begin{bmatrix} \dfrac{\partial F_1(\mathbf{x})}{\partial x_1} & \dfrac{\partial F_1(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial F_1(\mathbf{x})}{\partial x_n} \\ \dfrac{\partial F_2(\mathbf{x})}{\partial x_1} & \dfrac{\partial F_2(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial F_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial F_m(\mathbf{x})}{\partial x_1} & \dfrac{\partial F_m(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial F_m(\mathbf{x})}{\partial x_n} \end{bmatrix}_{m\times n}$$

9.3 Optimization

9.3.1 Introduction

The concept of optimization is well rooted as a principle underlying analysis of many complex decision problems. Precise optimization is an important tool in the analysis of physical systems. In order to make use of this tool, we identify some *objective*, a quantitative measure of the performance of the system under study. The objective depends on certain characteristics of the system called *variables*. Our goal is to find values of the variables that optimize the objective. Depending on whether the variables are restricted in a finite boundary or unrestricted the optimization models are characterized as *constrained* or *unconstrained* optimization techniques.

9.3.2 Mathematical formulation

Form mathematical point of view; optimization is the minimization or maximization of a object function

$$\text{Find } \hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, ..., \hat{x}_n]^T \text{ that minimizes } \mathbf{F}(\hat{\mathbf{x}}) \text{ subject to } c_i \tag{69}$$

- $\hat{\mathbf{x}}$ is the vector of *variables* also called *unknowns*
- $\mathbf{F}(\hat{\mathbf{x}})$ is the *objective function*
- c_i are *constraint functions*, which define certain constraints that the unknown vector $\hat{\mathbf{x}}$ must satisfy

This section reviews the mathematical motivation for optimization method and also details the algorithms. The emphasis of this analysis is on *unconstrained optimization* which is a common topic in mathematical programming.

9.3.3 Unconstrained optimization

By definition in unconstrained optimization, for a defined scalar function $E(\mathbf{x})$ where $E(\mathbf{x}) \in C^2$, we search for a vector $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ which minimizes $E(\mathbf{x})$.

A typical example of unconstrained optimization is the solution of linear system of equations:

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{12}x_2 + \cdots + a_{1N}x_N &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_2 + \cdots + a_{2N}x_N &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3N}x_N &= b_3 \\
&\vdots \\
a_{M1}x_1 + a_{M2}x_2 + a_{M2}x_2 + \cdots + a_{MN}x_N &= b_M
\end{aligned} \tag{70}$$

In matrix notation

$$\text{find } \hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, ..., \hat{x}_n]^T \qquad \text{that minimizes} \qquad E(\hat{\mathbf{x}}) = \left\| A\hat{\mathbf{x}} - b \right\|^2$$

where

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix} \quad , \quad b=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix} \tag{71}$$

By expanding the above equation we see that solving $A\mathbf{x}=b$ is equal to

find $\hat{\mathbf{x}}=[\hat{x}_1,\hat{x}_2,...,\hat{x}_n]^T$ that minimizes $E(\hat{\mathbf{x}})=\hat{\mathbf{x}}^{\mathrm{T}}\mathrm{A}^{\mathrm{T}}\mathrm{A}\hat{\mathbf{x}}-2\mathbf{b}^{\mathrm{T}}\mathrm{A}\hat{\mathbf{x}}+\mathbf{b}^{\mathrm{T}}\mathbf{b}$

9.3.4 Global and local optimization

The minimum or maximum point (extremum) can be either *global* or *local*. A global extremum is truly the highest or lowest function value whereas a local extremum is the highest or the lowest in a finite neighbourhood.

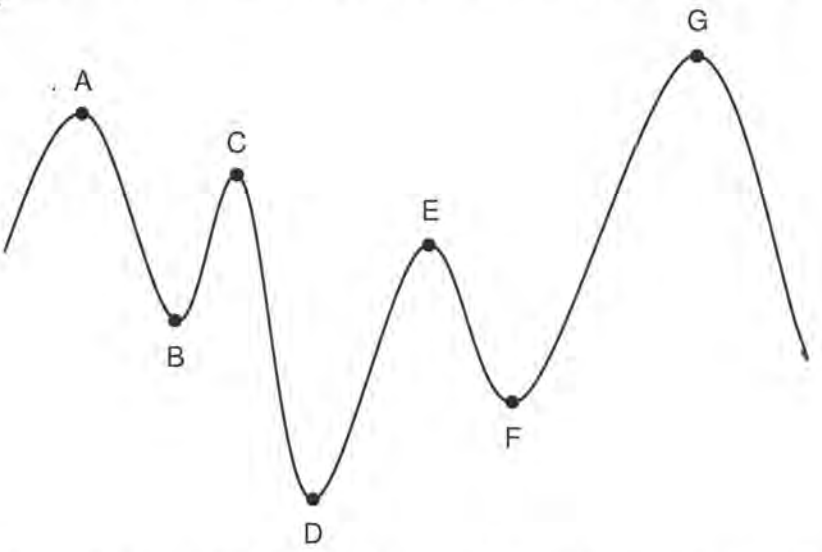


Figure 112 *Extrema of function in an interval, Points A, C and E are local maxima, but not global. The global maximum occurs at G. Points B and F are local minima, but not global. Point D is the global minimum.*

The task of minimization or maximization is trivially related and they are placed in a more general container referred as *optimization* problem.

9.4 Optimization Methodologies

Optimization methodologies fall into two main categories:

1. Linear Optimization
2. Non-Linear Optimization

As the name implies, a linear optimization problem is characterized by optimization of a linear function called objection function, whereas in nonlinear optimization, the objective function is a nonlinear function of independent variables. Figure 113 illustrates the hierarchy of different optimization domains and techniques.

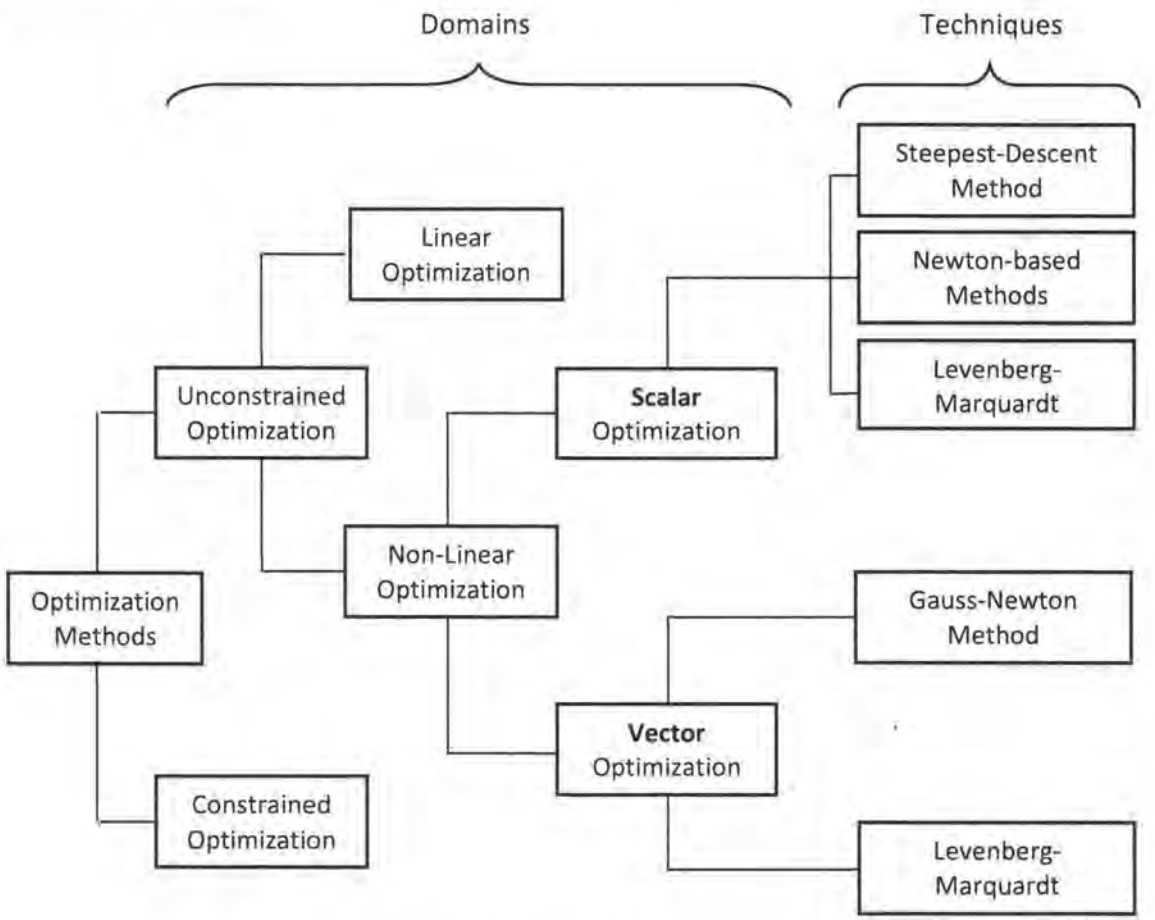


***Figure 113** Hierarchy of optimization domains and techniques*

9.4.1 Scalar optimization

Scalar optimization refers to the optimization methodologies that perform the optimization on single objective function and thus they are called Single Objective Optimization[33]. In this category we are given a scalar function $F(\mathbf{x}) = F(x_1, x_2, ..., x_n) \leftrightarrow F : R^n \rightarrow R$ and we want to find a vector $\hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, ..., \hat{x}_n]^T$ where $F(\hat{\mathbf{x}})$ takes on a minimum or maximum:

$$\text{minimization: } F(\hat{\mathbf{x}}) \leq F(\mathbf{x}) \tag{72}$$

[33] SOV

9.4.1.1 Steepest Descent Method (Gradient Method)

Steepest Descent Method is a simple **Line Search technique (LS)** that for a given objective function $F(\mathbf{x})$ and a guess point $\mathbf{x}^0$, it moves along the direction that is proportional to slope of the function at the guess point in negative direction $-F'(\mathbf{x}^0)$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha \cdot F'(\mathbf{x}^k) \tag{73}$$

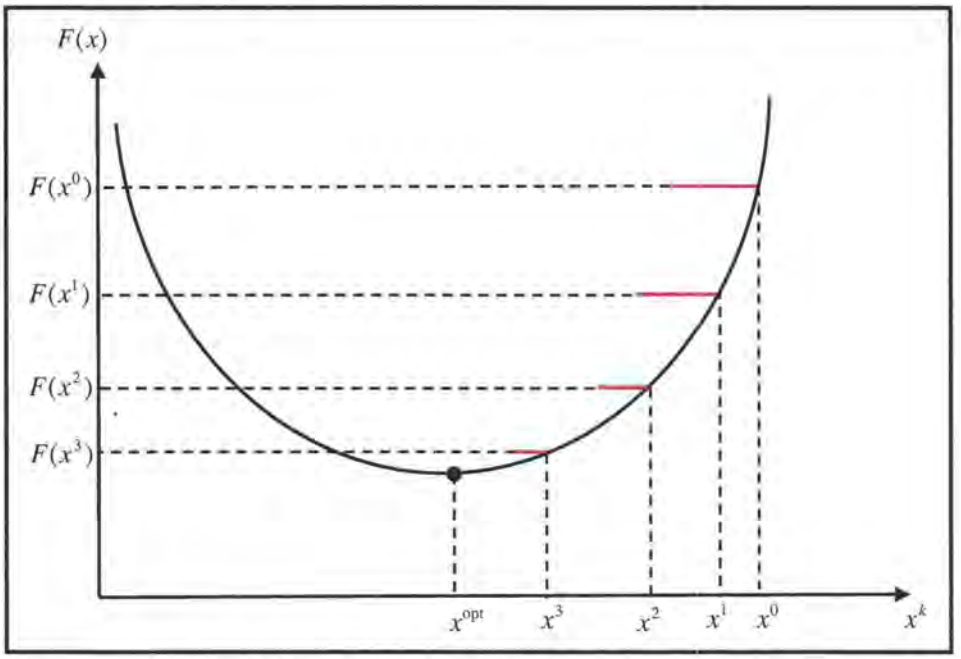


Figure 114 *The steepest descent in one dimension*

where α is the step size $\alpha \in (0,1)$. The step size could be chosen in variety of ways.For multidimensional variables, the steepest-descent method computes the search direction from

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha \cdot \nabla_x F(\mathbf{x}^k) \tag{74}$$

where $\nabla_x F(\mathbf{x}^k)$ is the gradient vector. The search direction $p^k = -\alpha \cdot \nabla_x F(\mathbf{x}^k)$ continues until $\mathbf{x}^k$ is the stationary point of function F .

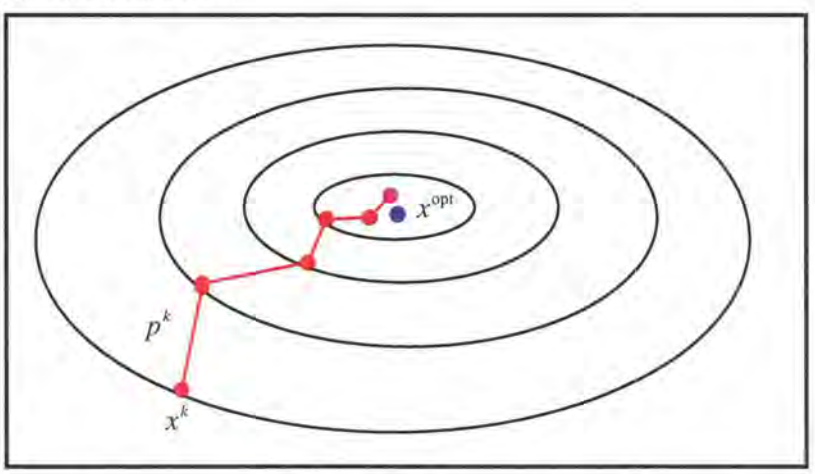


Figure 115 *The steepest Descent method for a function of two variables*

The advantage of the gradient descent method is that it only requires calculation of the gradient $\nabla F(x_k)$ not the second derivatives. The price of the simplicity is that the method is hopelessly efficient at solving most problems. From speed point of view, the other negative side of the steepest descent method is that it has a slow rate of convergence, since it converges at a linear rate (α) which is constant usually close to one. By comparison, Newton method provides faster rate of convergence at the expense of adding the higher derivatives' information to the equation system.

9.4.1.2 Newton-based methods

Newton method is the touchstone method for unconstrained problems and discussion of Newton-based methods is distributed almost in every optimization literature. The idea of Newton methodologies revolves around the use of Taylor's series expansion to approximate the objective function and solve for zero:

$$F(\mathbf{x}) = F(\mathbf{x}^k) + (\mathbf{x}-\mathbf{x}^k)\nabla_x F(\mathbf{x}^k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^k)^T \nabla^2_{xx} F(\mathbf{x}^k)(\mathbf{x}-\mathbf{x}^k) + h.o.t \qquad (75)$$

Depending on number of higher order terms used in the approximation of the objective function, the method is divided into first-order and second-order approximation.

1. First-order approximation

Newton's first order approximation is a well-known method for finding the **roots** of a real-valued function. In this application, Newton's methods can be known as *zero crossing method*. As the name implies the first order approximation is achieved by keeping the terms only to first order. The geometric interpretation of Newton's method in one-dimension corresponds to approximating the function F by a tangent line at point $(x^k, F(x^k))$. The intersection of the tangent line and x-axis determines the next iteration:

$$F(\mathbf{x}) = F(\mathbf{x}^k) + (\mathbf{x}-\mathbf{x}^k)\frac{\partial F(\mathbf{x}^k)}{\partial \mathbf{x}} \rightarrow F(\mathbf{x}^{k+1}) = 0 \rightarrow \mathbf{x}^{k+1} = \mathbf{x}^k - \frac{F(\mathbf{x}^k)}{\frac{\partial F(\mathbf{x}^k)}{\partial \mathbf{x}}} \qquad (76)$$

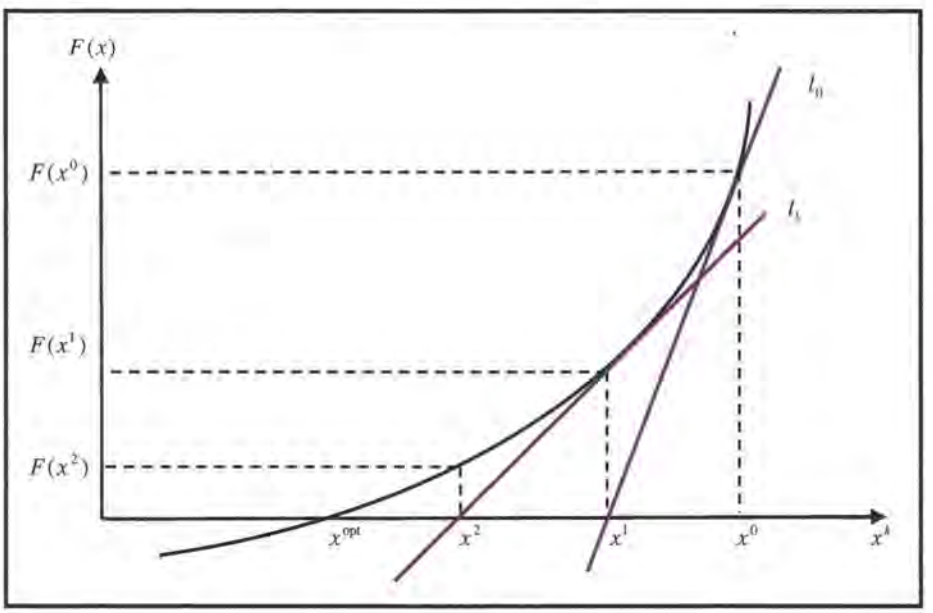


Figure 116 *Geometric illustration of Newton's method - first order approximation*

2. "Second-order approximation" (Inverse Hessian Method)

Alternatively, Newton's method can be applied for optimization problems. In this application, the derivative is zero at minimum or maximum therefore, the Newton's method could be applied to find the roots of the derivatives.

Similar to first order approximation, Newton's second order approximation also known as Inverse Hessian Method is derived by keeping second order terms and ignoring the rest terms:

$$F(\mathbf{x}) = F(\mathbf{x}^k) + (\mathbf{x}-\mathbf{x}^k)^T \nabla_x F(\mathbf{x}^k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^k)^T \nabla^2_{xx} F(\mathbf{x}^k)(\mathbf{x}-\mathbf{x}^k)$$

$$\frac{\partial F(\mathbf{x}^{k+1})}{\partial \mathbf{x}^T} = 0 \rightarrow \nabla_x F(\mathbf{x}^k) + \nabla^2_{xx} F(\mathbf{x}^k)(\mathbf{x}^{k+1}-\mathbf{x}^k) = 0$$

$$\underbrace{\mathbf{x}^{k+1}}_{[\,]_{n\times 1}} = \underbrace{\mathbf{x}^k}_{[\,]_{n\times 1}} - \underbrace{\underbrace{\left[\nabla^2_{xx} F(\mathbf{x}^k)\right]^{-1}}_{[\,]_{n\times n}}}_{\text{Hessian}} \underbrace{\underbrace{\nabla_x F(\mathbf{x}^k)}_{[\,]_{n\times 1}}}_{\text{Gradient}} \tag{77}$$

In one-dimension, Newton's method corresponds to approximating the function F by a tangent quadratic function at point $(x^k, F(x^k))$. The minimum point of the tangent curve determines the next iteration:

$$F(\mathbf{x}) = F(\mathbf{x}^k) + (\mathbf{x}-\mathbf{x}^k)\frac{\partial F(\mathbf{x}^k)}{\partial \mathbf{x}} + \frac{1}{2}(\mathbf{x}-\mathbf{x}^k)^2 \frac{\partial^2 F(\mathbf{x}^k)}{\partial \mathbf{x}^2}$$

$$\frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial F(\mathbf{x}^k)}{\partial \mathbf{x}} + (\mathbf{x}-\mathbf{x}^k)\frac{\partial^2 F(\mathbf{x}^k)}{\partial \mathbf{x}^2}$$

$$\frac{\partial F(\mathbf{x}^{k+1})}{\partial \mathbf{x}} = 0 \rightarrow \mathbf{x}^{k+1} = \mathbf{x}^k - \frac{\dfrac{\partial F(\mathbf{x}^k)}{\partial \mathbf{x}}}{\dfrac{\partial^2 F(\mathbf{x}^k)}{\partial \mathbf{x}^2}} \tag{78}$$

There are certain costs associated with Newton's approximation. In the classical Newton's method, it requires computation of second derivatives which is often expensive. For an *n*-variable problem, the Hessian matrix possesses $O(n^2)$ entities, so $O(n^2)$ expressions must be programmed to evaluate the derivatives and compute the Hessian Matrix. Such calculations are prohibitively subject to error that may cause the algorithm behave poorly or fail to converge. Furthermore, as *n* increases the cost of storage of the Hessian matrix increases rapidly.

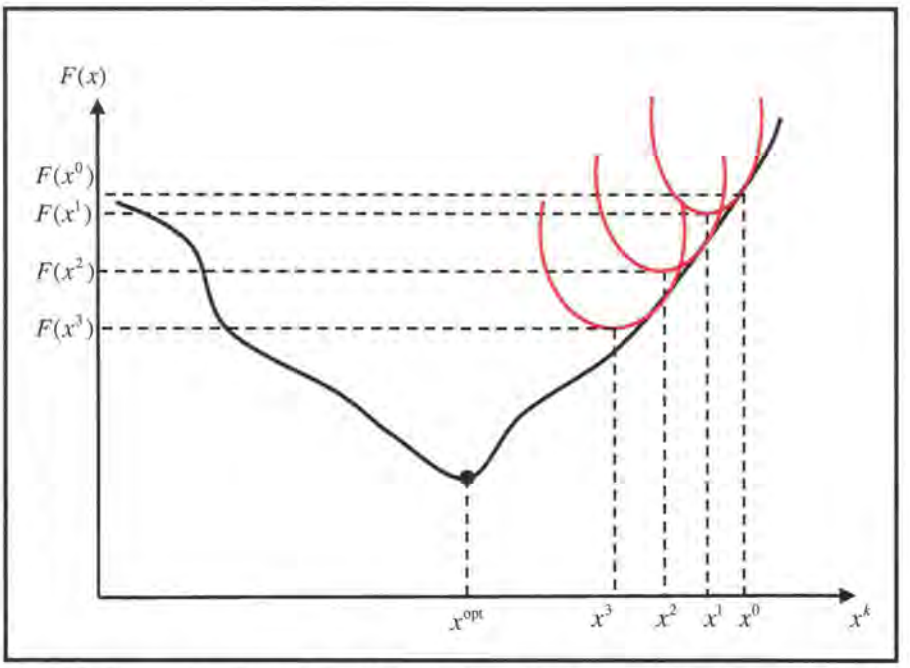


Figure 117 *Geometric illustration of Newton's method - second order approximation*

On the other hand, the algorithm does not come without compromises. The resulting technique has attractive convergence properties and slower rate of convergence. This is because Newton's method corrects the search direction

$$p^k = -\frac{\nabla_x F(x^k)}{\nabla^2_{xx} F(x^k)} \tag{79}$$

such that it always targets the minimum, whereas by comparison the gradient descent method points to the direction of maximum change, as shown in figure 118:

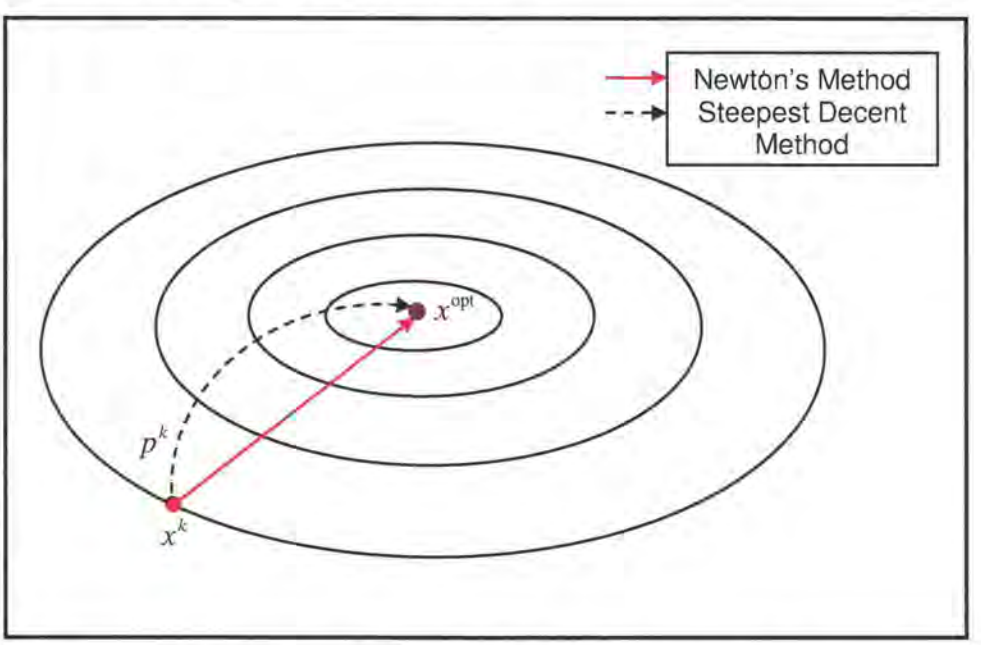


Figure 118 *Comparison of search direction between Newton's method and the steepest descent method*

9.4.1.3 The Levenberg-Marquardt method:

The Levenberg-Marquardt Algorithm (LMA) has become a standard method for nonlinear least square problem and widely adopted in variety of applications. Levenberg-Marquardt method is an elegant technique for varying smoothly between the extremes of Newton's Inverse Hessian Method and the Gradient Descend method. This section reviews the mathematical motivation for Levenberg-Marquardt Algorithm and details the algorithm.

Assume that the gradient vector is d where it stands for *derivatives* and H for the *Hessian* Matrix. Given our definition of the minimization algorithms and the new notations, the gradient descent algorithm can be reformulated as

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha \cdot \mathbf{d} \tag{80}$$

Similarly Newton's Inverse Hessian Method is

$$\mathbf{x}^{k+1} = \mathbf{x}^k - H^{-1} \cdot \mathbf{d} \tag{81}$$

Equation (80) and (81) differ only in the term that quantifies the slope, i.e. α and H^{-1}. Levenberg insight entails "blending" between these two extremes. He formulated his algorithm as follows where I is the identity matrix.

$$\mathbf{x}^{k+1} = \mathbf{x}^k - (H + \lambda I)^{-1}.\mathbf{d} \tag{82}$$

When the blending factor (sometime called damping factor) λ is small, the algorithm approaches the Inverse Hessian Method. If λ is large, the term λI is the significant term in comparison with H and the rule approaches the gradient descent method in which the step size is $\alpha = \frac{1}{\lambda}$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{\lambda} \cdot \mathbf{d} \tag{83}$$

Hence, λ can be set to **monitor the behaviour of the algorithm**. An intuition to adjust λ is, "If the error is decreasing, our approximation is likely working well and we expect that we are getting closed to the objection, so we should decrease λ to bring the algorithm close to Inverse Hessian Method. Conversely, if the error is increasing the approximation is not performing well and increase in λ can force the method more towards simple gradient method". The parameter can be changed by a factor of 10.

Marquardt improved this method in a smart incorporation of estimated Hessian Matrix H resulting in *Levenberg-Marquardt Algorithm*. His insight is that the components of the Hessian matrix give some information about the direction of convergence. So for example when λ is high and we are doing essentially gradient descent we should move *further* in the direction in which the gradient is small. Therefore we can use the diagonal of the Hessian instead of the Identity matrix:

$$\boxed{\mathbf{x}^{k+1} = \mathbf{x}^k - (H + \lambda.diag[H])^{-1}.\mathbf{d}} \tag{84}$$

9.4.2 Vector optimization

Vector optimization refers to the optimization methodologies that perform the optimization on multiple objective functions and thus they are sometimes called Multiple Objective Optimization[34].

In this category we are given a vector function

$$\mathbf{F}(\mathbf{x}) = [F_1(\mathbf{x}), F_2(\mathbf{x}), ..., F_m(\mathbf{x})]^T \tag{85}$$

and we want to find a vector $\hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, ..., \hat{x}_n]^T$ where the sum of the square of the deviations $S(\hat{\mathbf{x}})$ becomes minimal.

$$S(\hat{\mathbf{x}}) = \sum_{i=1}^{m} F_i^2(\hat{\mathbf{x}}) \tag{86}$$

9.4.2.1 Gauss-Newton algorithm

One of the commonly used methods in nonlinear minimizations problems is Gauss-Newton algorithm. In this section, we derive Gauss-Newton's method for a system of nonlinear equations. Suppose $\mathbf{F}: R^n \rightarrow R^m$ for a set of m nonlinear equations in n variables. The basic problem studied here is the solution to the system of nonlinear equations:

$$\text{given } \mathbf{F}: R^n \rightarrow R^m, \text{ find } \hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, ..., \hat{x}_n]^T \text{ such that } \mathbf{F}(\hat{\mathbf{x}}) = 0 \tag{87}$$

where F is assumed to be continuously differentiable.

In large-scale optimization, the first order Taylor expansion of a vector function is given by:

$$\underbrace{\mathbf{F}(\mathbf{x})}_{[\,]_{m\times 1}} = \underbrace{\mathbf{F}(\mathbf{x}^k)}_{[\,]_{m\times 1}} + \underbrace{\mathbf{J}(\mathbf{x}^k)}_{[\,]_{m\times n}} \underbrace{(\mathbf{x}-\mathbf{x}^k)}_{[\,]_{n\times 1}} \tag{88}$$

Jacobian

where $\mathbf{J}(\mathbf{x})$ (the Jacobian) is the first partial derivative of $\mathbf{F}(\mathbf{x})$ given by:

$$\boldsymbol{J}_{m\times n}(\mathbf{x}) = \begin{bmatrix} \frac{\partial F_1(\mathbf{x})}{\partial x_1} & \frac{\partial F_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial F_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial F_2(\mathbf{x})}{\partial x_1} & \frac{\partial F_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial F_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m(\mathbf{x})}{\partial x_1} & \frac{\partial F_m(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial F_m(\mathbf{x})}{\partial x_n} \end{bmatrix}_{m\times n} \tag{89}$$

If we let $\mathbf{x}^{k+1}$ be the $\mathbf{x}$ that gives $\mathbf{F}(\mathbf{x}^{k+1}) = 0$, the following expression is obtained:

$$\mathbf{F}(\mathbf{x}^k) + \boldsymbol{J}_{m\times n}(\mathbf{x}) \quad (\mathbf{x}^{k+1} - \mathbf{x}^k) = 0 \rightarrow \quad \boldsymbol{J}_{m\times n}(\mathbf{x}) \quad (\mathbf{x}^{k+1} - \mathbf{x}^k) = -\mathbf{F}(\mathbf{x}^k) \tag{90}$$

To find an iterative expression for $\mathbf{x}^{k+1}$, we need to find the inverse of the Jacobian matrix. Jacobian matrix in its general size is not invertible because it's a non-square matrix. In order to make it

[34] MOV

invertible, we transform it into a square matrix by multiplying the sides by the transpose of the Jacobian matrix as follows:

$$(\boldsymbol{J}(\mathbf{x}))^T.\boldsymbol{J}(\mathbf{x}) \quad (\mathbf{x}^{k+1}-\mathbf{x}^k) = -(\boldsymbol{J}(\mathbf{x}))^T.\mathbf{F}(\mathbf{x}^k)) \tag{91}$$

$$\rightarrow \underbrace{(\mathbf{x}^{k+1}-\mathbf{x}^k)}_{[\,]_{n\times 1}} = -\underbrace{\left(\boldsymbol{J}^T(\mathbf{x}).\boldsymbol{J}(\mathbf{x})\right)^{-1}}_{[\,]_{n\times n}} \underbrace{(\boldsymbol{J}(\mathbf{x}))^T}_{[\,]_{n\times m}} . \underbrace{\mathbf{F}(\mathbf{x}^k)}_{[\,]_{m\times 1}}$$

Hence, the final expression is obtained as:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \left(\boldsymbol{J}^T(\mathbf{x}).\boldsymbol{J}(\mathbf{x})\right)^{-1}(\boldsymbol{J}(\mathbf{x}))^T.\mathbf{F}(\mathbf{x}^k) \tag{92}$$

9.4.2.2 Levenberg-Marquardt algorithm

Given $\mathbf{F}(\mathbf{x})$ and $\hat{\mathbf{F}}(\mathbf{x}) = \mathbf{F}(\mathbf{x}^k) + \mathbf{J}(\mathbf{x}^k)(\mathbf{x}-\mathbf{x}^k)$, the Gauss-Newton method finds the solution to the problem : "Find the $\mathbf{x}$ that minimizes $\left\|\hat{\mathbf{F}}(\mathbf{x})\right\|$". The solution is given by the well-known normal equation:

$$\underbrace{\mathbf{x}-\mathbf{x}^k}_{\delta_p} = -\left(\boldsymbol{J}^T(\mathbf{x}).\boldsymbol{J}(\mathbf{x})\right)^{-1}(\boldsymbol{J}(\mathbf{x}))^T.\mathbf{F}(\mathbf{x}) \tag{93}$$

Levenberg's contribution is to regularize the normal equation of the Gauss-Newton algorithm by adding a diagonal component to the $\boldsymbol{J}^T\boldsymbol{J}$ matrix:

$$(\boldsymbol{J}^T\boldsymbol{J} + \lambda\mathbf{I})\delta_p = \boldsymbol{J}^T(-\mathbf{F}(\mathbf{x})) \tag{94}$$

where λ is a small non-negative value that could be varied during iteration.

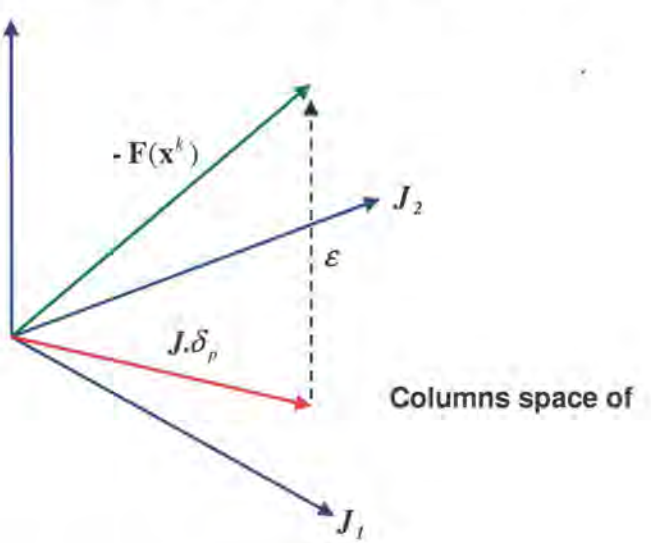


Figure 119 *Geometry of least square problem in 3 dimensions*

10 Appendix C "Homogenous Coordinate System"

10.1 Affine transformation

For most computer graphics practitioners, it is very crucial to master transforms. With correct transforms in hand, you can position, reshape and animate objects, lights and cameras. There are only a few operations that can be performed with transforms, but they are sufficient to demonstrate the importance of the transforms' role in real-time graphics or in any kind of computer graphics.

A transformation is a function that takes a point or vector and maps into another point or vector. Such a function can be illustrated by looking at the figure below:

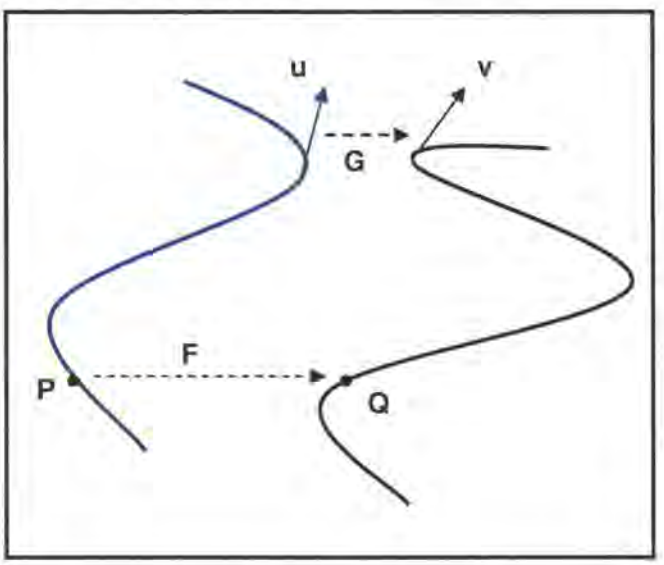


Figure 120 *Transformation*

or by writing down the functional form

$$Q = F(P) \tag{95}$$

for points ,or

$$v = G(u) \tag{96}$$

for vectors.

This formulation is too general to be useful, as it encompasses all single-valued mapping of points and vectors. We need to obtain a formulation that simplifies the usage of above relations. A useful class of transformations however is obtained if linearity restriction is placed place on f . By definition, a **linear transformation** (linear function) is a function that for any scalars α and β , and any vertices p and q ,

$$f(\alpha p + \beta q) = af(p) + \beta f(q) \tag{97}$$

Rotation is an example of a linear transformation. The importance of such a function is that if we know the transformation of vertices, we can obtain the transformation of linear combinations of vertices by linear combination of transformation of vertices. In other words, we do not need to revaluate the transformation for every linear combination. An **affine transformation** is composed of a linear transformation and a translation (or "shift").

10.2 Homogeneous coordinate system

10.2.1 Why a new coordinate system?

10.2.1.1 To distinguish between points and vectors

In some situations we may need to differentiate between points and vectors. In conventional 3-dimenisonal coordinate systems both objects are represented with the same notation (x, y, z) and therefore for given coordinates the differentiation between points and vectors is not possible.

In order to distinguish between the two quantities we need a general method to identify these terms. In an affine space, once we fix a particular reference point - the origin- we can represent all points unambiguously. This representation however, requires us to know both the reference point and the basis vectors. The origin and the basis vectors determine a **frame**. Loosely, this extension fixes the origin of the vector coordinate system at some point P_0. Within a given frame, every vector can be written uniquely as

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 \tag{98}$$

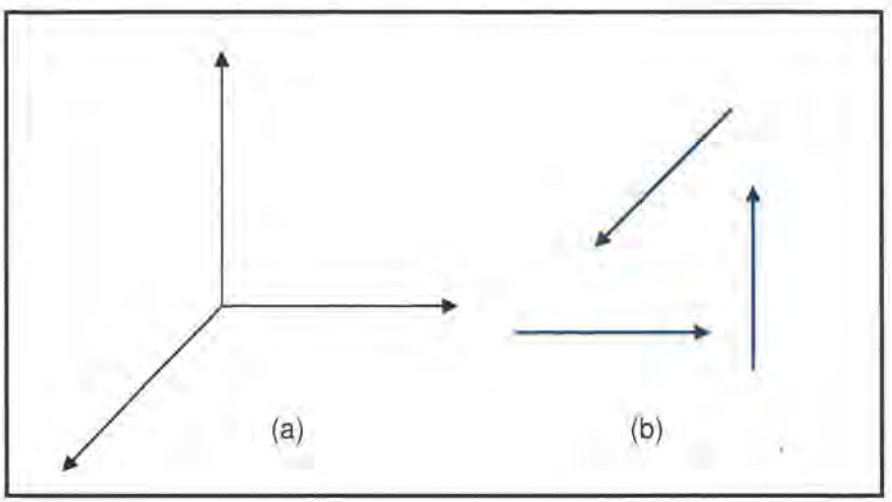


Figure 121 *Coordinate systems Vectors originating from a common point (a) Vectors moved (b)*

In addition every point can be written uniquely as

$$P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3 \tag{99}$$

Thus, the representation of a particular vector in a frame requires three scalars; the representation of a point requires three scalars and the knowledge of where the origin is located. As we shall see in the next section, in **homogenous coordinate system** we avoid the difficulties caused by vectors having magnitude and direction but no fixed position.

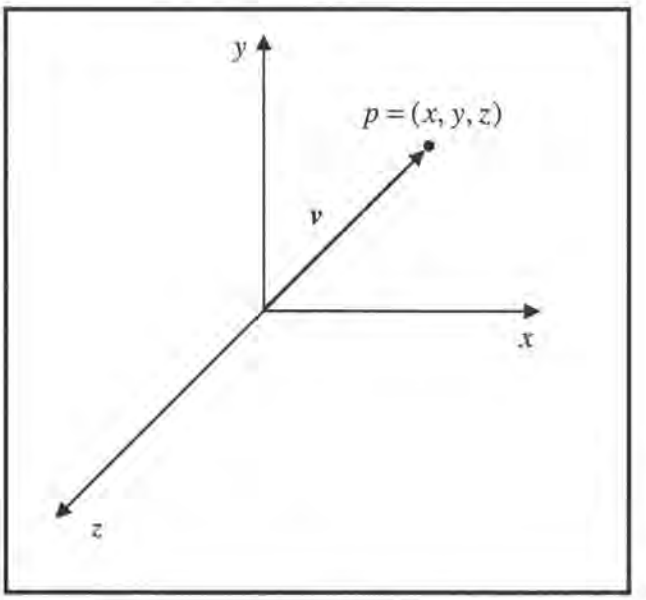


Figure 122 *A bad representation of a vector*

Instead we are able to represent points and vectors in a manner that will allow us to use matrix representation and at the same time maintain a distinction between two different geometric objects.

10.2.1.2 To concatenate the transformations in terms of simple matrix multiplication

In a homogenous coordinate system different transformation types have the same size of 4×4 .For example rotation and translation transformations are defined as:

$$\boldsymbol{T}(t)=\boldsymbol{T}(t_x,t_y,t_z)=\begin{bmatrix}1&0&0&t_x\\0&1&0&t_y\\0&0&1&t_z\\0&0&0&1\end{bmatrix} \tag{100}$$

$$\boldsymbol{R}_x(p)=\begin{pmatrix}1&0&0&0\\0&\cos(p)&-\sin(p)&0\\0&\sin(p)&\cos(p)&0\\0&0&0&1\end{pmatrix} \tag{101}$$

$$\boldsymbol{R}_y(h)=\begin{pmatrix}\cos(h)&0&\sin(h)&0\\0&1&0&0\\-\sin(h)&0&\cos(h)&0\\0&0&0&1\end{pmatrix} \tag{102}$$

$$R_z(r) = \begin{pmatrix} \cos(r) & -\sin(r) & 0 & 0 \\ \sin(r) & \cos(r) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (103)$$

Thus, we can create examples of affine Transformation by multiplying transformations together, or **concatenating** sequences of basic transformations.

Figure 123 *Concatenation of transformations one at a time*

Suppose that we perform successive transformations on a point p, creating a new point q; since the matrix product is associative we can write:

$$q = CBAP \quad (104)$$

However the order that we carry out transformation might affect the efficiency of the calculations. In one view, we may carry out the transformations one at a time **A,** followed by **B**, followed by **C**, as shown in figure 123:

$$q = (C(B(Ap))) \quad (105)$$

This order is efficient when we are to transform a single point, because each matrix multiplication involves multiplying a columns vector by a square matrix.

If on the other hand, we have many points to transform, we can carry on in two steps. First we evaluate the **compound matrix** of transformations:

$$M = CBA \quad (106)$$

Then we use this matrix on each point:

$$q = Mp \quad (107)$$

This order corresponds to the structure as illustrated in figure 124, here we compute **M** first then load it into a transformation unit.

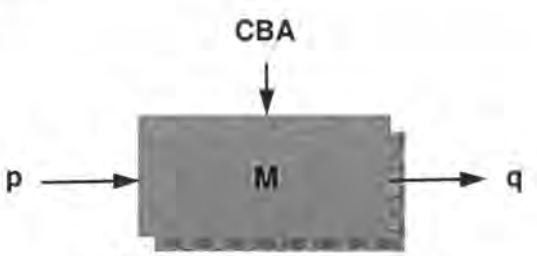


Figure 124 *Transformation*

Although we do a little more calculations in computing **M**, the number count operations could be extremely reduced. Since **M** may be applied to tens of thousands of points which means the extra works significantly saved by using a single matrix multiplication for each point.

To illustrate this let's consider a simple example. Consider figure 123 and figure 124 where we would like to perform three successive transformations on a set of points $\{p_i\}_{i=1}^{M}$ to obtain point the set $\{q_i\}_{i=1}^{M}$ where M is the number of points the transformation is performed on. Assume the transformation matrices are of size 4×4 . We want to find the number of operations required in each case.

If $\mathrm{M}=1$:

Number of Operations for case 1 (three-step operation):

$$\mathrm{N}=(4\times4\times1)+(4\times4\times1)+(4\times4\times1)=3\times4^2=48$$

Number of Operations for case 2 (one-step operation):

$$\mathrm{N}=(4\times4\times4)+(4\times4\times1)=128+16=144$$

If $\mathrm{M}=\mathrm{k}$:

Number of Operations for case 1 (three-step operation):

$$\mathrm{N}=(4\times4\times1)+(4\times4\times1)+(4\times4\times1)+\cdots$$
$$+(4\times4\times1)+(4\times4\times1)+(4\times4\times1)=48k$$

Number of Operations for case 2 (one-step operation):

$$\mathrm{N}=\underbrace{(4\times4\times4)}_{128}+\underbrace{(4\times4\times1)}_{1}+\cdots+\underbrace{(4\times4\times1)}_{k}=128+16\mathrm{k}$$

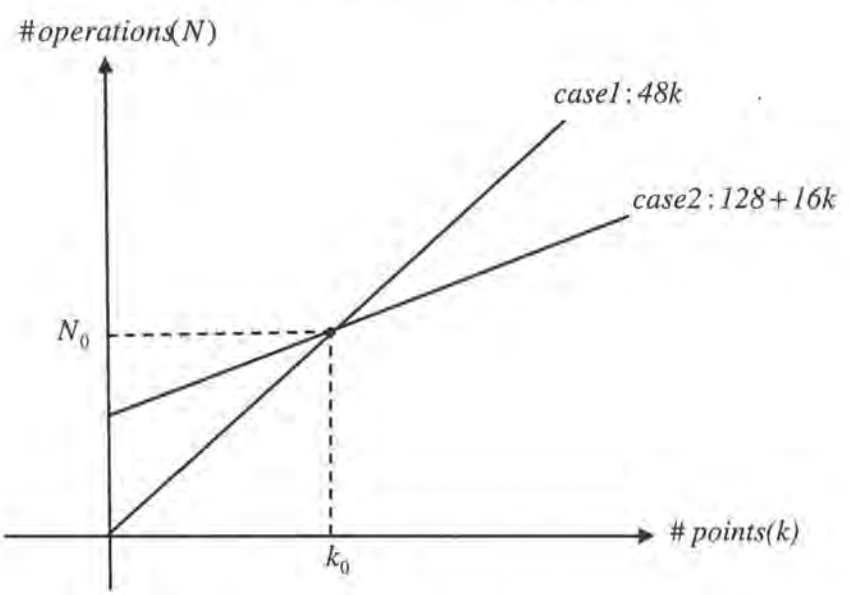


Figure 125 *Comparison of number of operations for two cases*

As it could be seen from the figures above, for large values of k, the number of operations required in case 2 is less than case 1 and as N grows, the gap between the number of operations in two cases increases. The breakeven point is $k_0 = 4$:

$$128+16k_0 = 48k_0 \rightarrow k_0 = 4 \tag{108}$$

10.2.2 Definition of homogenous coordinate system

We wish is to formulate a point P located at (x, y, z) using a three-dimensional frame defined in terms of basis vectors (v_1, v_2, v_3) and a reference point P_0 so that

$$P = P_0 + xv_1 + yv_2 + zv_3 \tag{109}$$

When this representation is used, it has the same form as the vector that is

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 \tag{110}$$

and the representation of w is the column matrix

$$w = [\alpha_1 \quad \alpha_2 \quad \alpha_3]^T \tag{111}$$

This representation could sometimes cause confusion because many references associate the point (x, y, z) with vector defined in terms of the line passing through the origin to this point.

For example, the vector from the point $(2,2,2)$ to $(5,4,3)$ is

$$\begin{aligned} &w = (5-2)v_1 + (4-2)v_2 + (3-2)v_3 \rightarrow w = 3v_1 + 2v_2 + 1v_3 \\ &\rightarrow w = [3 \quad 2 \quad 1]^T \end{aligned} \tag{112}$$

Similarly the point P with coordinates $(3,2,1)$ and reference point $P_0 = (0,0,0)$ is

$$P = P_0 + 3v_1 + 2v_2 + 1v_3 \rightarrow P = 3v_1 + 2v_2 + 1v_3 \tag{113}$$

Four-dimensional column vectors can be used to represent **both** points and vectors in three dimensions. In the frame recognised by (v_1, v_2, v_3, P_0), any point can be written uniquely as:

$$P = \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3 + P_0 \tag{114}$$

We can formulate this relation in terms of matrix product, as

$$P = [v_1 \quad v_2 \quad v_3 \quad P_0] \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ 1 \end{bmatrix} \tag{115}$$

This expression is not an inner product, because the elements of the matrices are dissimilar; nevertheless, the expression is evaluated as if it were an inner product by multiplying the corresponding elements and summing the results.

The four-dimensional row vector on the right-side if the equation above is the homogenous-coordinate representation of the point P in the frame defined by v_1, v_2, v_3 and P_0:

$$P = [\beta_1 \quad \beta_2 \quad \beta_3 \quad 1]^T \tag{116}$$

In the same frame, any vector w can be written as

$$\begin{aligned} w &= \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 \\ &= [v_1 \quad v_2 \quad v_3 \quad P_0] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 0 \end{bmatrix} \end{aligned} \tag{117}$$

Hence the vector w in the homogenous coordinate system is represented by

$$w = [\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad 0] \tag{118}$$

There are numerous advantages to using the homogenous coordinate system. For instance, we can simply note that we can carry out operations on points and vectors just by using ordinary matrix algebra. Similarly, in many computer vision tasks, it is required to investigate the relationship between changes of frames in two coordinate systems (two frames) as frames defined like below:

- Frame 1: (v_1, v_2, v_3, P_0)
- Frame 2: (u_1, u_2, u_3, Q_0)

With a homogenous-coordinate representation in place, we can express the basis vector and reference point of the second frame in terms of the first as:'

$$\begin{cases} u_1 = \lambda_{11} v_1 + \lambda_{12} v_2 + \lambda_{13} v_3 \\ u_2 = \lambda_{21} v_1 + \lambda_{22} v_2 + \lambda_{23} v_3 \\ u_3 = \lambda_{31} v_1 + \lambda_{32} v_2 + \lambda_{33} v_3 \\ Q_0 = \lambda_{41} v_1 + \lambda_{42} v_2 + \lambda_{43} v_3 + P_0 \end{cases} \tag{119}$$

In matrix notation, the equations can be written as:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ Q_0 \end{bmatrix} = \mathbf{M} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} \quad , \quad \mathbf{M} = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \lambda_{13} & 0 \\ \lambda_{21} & \lambda_{22} & \lambda_{23} & 0 \\ \lambda_{31} & \lambda_{32} & \lambda_{33} & 0 \\ \lambda_{41} & \lambda_{42} & \lambda_{43} & 1 \end{bmatrix} \tag{120}$$

$\mathbf{M}$ is called the **matrix representation** of the changes of the frames. The matrix $\mathbf{M}_{4\times4}$ could be used to compute the changes between two frames; In other words, given points **or** vectors $\mathbf{a}$ and $\mathbf{b}$ in two coordinate systems we could write their relationship as:

$$\mathbf{b}^{\mathrm{T}}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ Q_0 \end{bmatrix} = \mathbf{b}^{\mathrm{T}}\mathbf{M}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} = \mathbf{a}^{\mathrm{T}}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} \tag{121}$$

Therefore,

$$\mathbf{a} = \mathbf{M}^{\mathrm{T}}\mathbf{b} \tag{122}$$

10.2.3 Affine transformations in homogenous coordinate system:

A linear transformation can be formulated in terms of two representations u and v as a matrix multiplication:

$$v = Au \tag{123}$$

where A is a square 4×4 matrix and is of form

$$A = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{124}$$

The matrix above has 12 unknown parameters and we say this 3D transformation has *12 degrees of freedom.* However, since in homogonous coordinate systems vectors are represented as

$$u = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 0 \end{bmatrix} \tag{125}$$

they use only 9 degrees of freedom in the transformation of vectors. Whereas, points have the full 12 degrees of freedom.

$$v = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ 1 \end{bmatrix} \tag{126}$$

These results are important in many practical graphics. For example, we know affine transformations preserve lines; Knowledge of this quality helps to transform the homogenous representation of the end points of a line segment to determine complete a transformed line. Therefore, we can implement our graphics system as a pipeline that passes end points through transformation and we can finally construct the line at the next stage.

11 References

[1] Angel, E. (2002). *Interactive Computer Graphics : A top-down Approach Using OpenGL* (Third uppl.). Addison-Wesley.

[2] Baratoff, G., & Blanksteen, S. (u.d.). *Tracking Devices.* Hämtat från http://www.hitl.washington.edu/scivw/EVE/I.D.1.b.TrackingDevices.html den 10 June 2009

[3] Barsky, B. A. (2001). *Computer Animation: Algorithms and Techniques (The Morgan Kaufmann Series in Computer Graphics).* (M. Kaufmann, Red.) Berkeley: Univeristy of California.

[4] Begault, D. R. (1994). *3-D Sound for Virtual Reality and Multimedia.* San Diego, CA, USA: Academic Press Professional, Inc.

[5] Blauert, J. (1999). *Spatial Hearing - The Psychophysics of Human Sound Localization.* Massachusetts, USA: The MIT Press.

[6] Choi, C., Baek, S.-M., & Lee, S. (2008). Real-time 3D Object Pose Estimation and Tracking for Natural Landmark Based Visual Servo. *International Conference on Intelligent Robots and Systems Acropolis Convention Center* (ss. 22-26). Nice: IEEE/RSJ.

[7] DeMenthon, D. F., & Davis, L. S. (1992). Model-Based Object Pose in 25 Lines of Code. *International Journal of Computer Vision archive , 588* (1-2), 335 - 343.

[8] Dornaika, F., & Garcia, C. (1999). Pose Estimation using Point and Line Correspondences. *Real-Time Imaging , 5* (3), 215-230(16).

[9] Evans, M. J., Tew, A. I., & Angus, J. A. *SPATIAL AUDIO TELECONFERENCING - WHICH WAY IS BETTER?* York: University of York, Department of Electronics.

[10] Gardner, W. G. (1999). 3D Audio and Acoustic Environment Modeling. *Wave Arts* .

[11] Gareth, L., Eun-jung, H., & Robyn, O. (2000). A 3D Head Tracker for an Automatic Lipreading System. *In Proc. Australian Conf. on Robotics and Automation (ACRA2000). 1*, ss. 37-42. Melbourne: Australian Robotics & Automation Association.

[12] Haralick, B. M., Lee, C.-N., Ottenberg, K., & Nölle, M. (1991). Analysis and Solutions of the Three Point Perspective Pose Estimation Problem. *IEEE Computer Society Conference on In Computer Vision and Pattern Recognition* (ss. 592-598). Hamburg: Universitaet Hamburg .

[13] Huang, T., Blostein, S., & Margerum, E. (1986). Least-squares estimation of motion parameters from 3D point correspondence. *IEEE Conf. Computer Vision and Pattern Recogntition* .

[14] Jot, J.-M., & A.Chaigne. (February 1991). Digital delay networks for designing artificial reverberators. *Audio Engineering Society Preprint 3030 (E-2)* , 19-22.

[15] K.S.Arun, Huang, T., & Blostein, S. (1987). Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence , 9* (5), 698-700.

[16] Kreylos, O. (u.d.). *Wiimote hacking.* Hämtat från http://graphics.cs.ucdavis.edu/~okreylos/ResDev/Wiimote/index.html den 15 March 2009

[17] Kreylos, O. *With Algebra and a Wii Remote You Can Virtually Rule the World.* Drake University.

[18] Krik, A. G., O'Brien, J. F., & Forsyth, D. A. (2005). Skeletal Parameter Estimation from Optical Motion Capture Data. *roceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). 2*, ss. 782-788. California: University of San Deigo.

[19] Lee, J. C. (2008). Hacking the Nintendo Wii Remote. *IEEE* .

[20] Lee, J. (u.d.). *Wii remote projects.* Hämtat från http://johnnylee.net/projects/wii/ den 10 December 2008

[21] Li-Juan, Q., Yu-Lan, H., Ying-Zi, W., Hong, W., & Yue, Z. (2008). Research on Optimum Position for Straight Lines Model. *Springer , 5226/2008*, 521-528.

[22] Lourakis, M. I. (2005). *A Brief Description of the Levenberg-Marquardt Algorithm Implemented.* Crete: Institute of Computer Science,Foundation for Research and Technology - Hellas (FORTH),Greece.

[23] Ludwig, L. F., Pincever, N., & Cohen, M. (1990). Extending the Notion of a Window System to Audio. *23* (8), 66 - 72 .

[24] Medioni, G., & Kang, S. B. (2004). *Emerging Topics in Computer Vision.* NJ, USA: Prentice Hall PTR; Har/DVD edition.

[25] Moorer, J. (1979). About This Reverberation Business. *Computer Music , 3* (2), 13-28.

[26] *Motion Capture.* (u.d.). Hämtat från Wikipedia: http://en.wikipedia.org/wiki/Motion_capture den 15 5 2009

[27] Nash, S. G., & Sofer, A. (1995). *Linear and Nonlinear Programming.* McGraw-Hill Science/Engineering/Math.
[28] Nocedal, J., & Wright, S. J. (2000). *Numerical Optimization* (2nd uppl.). Springer.
[29] Ohayon, S., & Rivlin, E. (2006). Robust 3D Head Tracking Using Camera Pose Estimation. *International Conference on Pattern Recognition (ICPR). 1*, ss. 1063-1066. Washington, DC: IEEE Computer Society.
[30] Or, L. W., Luk, W. S., Wong, K. H., & King, I. (1998). An Efficient Iterative Pose Estimation Algorithm. *Proceedings of the Third Asian Conference on Computer Vision - Volume II. 1352* , ss. 559 - 566. London: Springer-Verlag.
[31] P.Sandgren. (u.d.). Early Reverberation Effect for the R15 Platform. *EriDoc* .
[32] Paavola, M., Karlsson, E., & Page, J. (den 31 May 2005). 3D Audio for Mobile Devices via Java.
[33] Qin, L., Hu, Y., Wei, Y., Wang, H., & Zhou, Y. (2008). New Algorithm for Determining Object Attitude Based on Trapezoid Feature. *Springer , 5226*, 405-413.
[34] Quan, L., & Lan, Z. (1999). Linear N-Point Camera Pose Determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence , 21* (8), 774-780.
[35] Ragab, M., & Wong, K. (2007). *Extended Kalman Filter Based Pose Estimation Using Multiple Cameras.* The Chinese University of Hong Kong, Department of Information Engineering. Hong Kong: Internal report of the CSE Dept.
[36] Ranta, C., & McGowan, S. (den 22 May 2003). Human Interface Device (HID) Profile.
[37] *Report on Research.* (u.d.). Hämtat från http://computing.ornl.gov/internships/rams/archive/rams05/websites05/j_dacunha/research_paper.htm 2009
[38] Roweis, S. *Levenberg-Marquardt Optimization.* Torento.
[39] Rumsey, F. (2001). *Spatial Audio.* Oxford, Uk: Focal Press.
[40] Shapiro, L. G., Stockman, G. C., & Shapiro, L. G. (2001). *Computer Vision.* Prentice Hall.
[41] Smit, J. (2008). *Affordable Head Tracking.* Univeristy of Groningen, Computer Science.
[42] Staunter, J., & Puckette, H. (1982). Designing multi-channel reverberators. *Computer Music , 6(1).*
[43] Trucco, & Verri, A. (1998). *Introductory Techniques for 3-D Computer Vision.* NJ, USA: Prentice Hall.
[44] Umeyama, S. (1991). Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence , 13* (4), 376 - 380 .
[45] Y.Tsai, R. (1987). A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. *IEEE Journal of Robotics and Automation* , 323-344.
[46] Yuan, J. S.-C. (1989). A General Photogrammetric Method for Determining Object Position and Orientation. *IEEE Transactions on Robotics and Automation , 5* (2), 129-142.