DESIGN AND IMPLEMENTATION OF A HEAD TRACKING CONTROLLED PAN AND TILT VISION SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HASAN ÖLMEZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

AUGUST 2013

Approval of the thesis:

DESIGN AND IMPLEMENTATION OF A HEAD TRACKING CONTROLLED PAN AND TILT VISION SYSTEM

submitted by **Hasan ÖLMEZ** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** ____________________

Prof. Dr. Süha Oral
Head of Department, **Mechanical Engineering** ____________________

Assoc. Prof. Dr. İlhan Konukseven
Supervisor, **Mechanical Engineering Dept., METU** ____________________

Asst. Prof. Dr. Buğra Koku
Co-Supervisor, **Mechanical Engineering Dept., METU** ____________________

Examining Committee Members:

Asst. Prof. Dr. Yiğit Yazıcıoğlu
Mechanical Engineering Dept., METU ____________________

Assoc. Prof. Dr. İlhan Konukseven
Mechanical Engineering Dept., METU ____________________

Asst. Prof. Dr. Buğra Koku
Mechanical Engineering Dept., METU ____________________

Asst. Prof. Dr. Kıvanç Azgın
Mechanical Engineering Dept., METU ____________________

Asst. Prof. Dr. Andaç Töre Şamiloğlu
Mechanical Engineering Dept., Baskent University ____________________

Date: 29.08.2013

ABSTRACT

DESIGN AND IMPLEMENTATION OF A HEAD TRACKING CONTROLLED PAN AND TILT VISION SYSTEM

Ölmez, Hasan
M.Sc., Department of Mechanical Engineering
Supervisor : Assoc. Prof. Dr. İlhan Konukseven
Co-Supervisor : Asst. Prof. Dr. Buğra Koku

August 2013, 89 pages

Head tracking applications are getting widely used in robotics implementations such as in military industry for surveillance applications of remote controlled unmanned vehicles, in medical applications for heart and vessel surgeries, in entertainment and game applications for virtual reality. It can be used for helping the operator of teleoperated unmanned vehicle with improving his performance by decreasing workload and managing stress. Head tracking is a challenging application due to real time tracking and synchronization problem. Considering this challenging, in this thesis head tracking controlled pan, tilt and roll vision system is accomplished. Head tracking is achieved with an orientation sensor which has 3-axis accelerometer, angular rate sensor, and compass sensor modules. With these modules the sensor can measure orientation in all three dimensions. This sensor is fixed on a polystyrene helmet which enables operator to fix the sensor on his head. With the motion of the operator's head, the position change is tracked. This tracked data is used to control pan-tilt-roll mechanism that a 3D video camera fixed on it. From this video camera a captured video is transferred to a head mounted display on 2D-3D and/or on a 6 LED monitors display system on 2D. At the end of the thesis, for validating the system two different experiments are made, one of them for the differences between 2D and 3D vision on HMD and the other is for testing 6 LED monitors display and HMD with 2D vision. These tests are done for looking the performance of operators on different displays (HMD and 6 LED monitors) and on vision types (2D-3D).

Keywords: Orientation sensor, head tracking, servo motor, pan-tilt-roll mechanism, 3D vision, video camera, head mounted display, LED monitor, vision system

ÖZ

KAFA TAKİBİ SİSTEMİ İLE KONTROL EDİLEBİLEN PAN-TILT KAMERA SİSTEMİNİN MODELLENMESİ

Ölmez, Hasan
Yüksek Lisans, Makine Mühendisliği Bölümü
Tez yöneticisi : Assoc. Prof. Dr. İlhan Konukseven
Ortak tez yöneticisi : Asst. Prof. Dr. Buğra Koku

Ağustos 2013, 89 sayfa

Kafa takip sistemleri günümüzde robotik uygulamalarda ve özellikle sanal gerçeklik uygulamalarında son zamanlarda sıkça kullanılmaya başlayan bir sistemdir. Askeri alanlardan oyun, eğlence uygulamalarına kadar pek çok alanda yaygınlaşmaya ve ihtiyaca göre farklı gereksinim ve tasarımlarla son kullanıcıya sunulmaktadır. Askeri alanlarda uzaktan kontrol edilebilen insansız araçlarda, tıpta kalp ve damar ameliyatlarında sıkça kullanılmaktadır. Uzaktan kontrol edilebilen insansız araçlarda kullanıcının kullanım performansını artırabilmek için kafa takip sistemleri sıkça kullanılmaktadır. Kafa takibi gerçek zamanlı pozisyon kontrolü ve eşzamanlı hareket sağlayamama problemleri sebebi ile zor bir çalışma alanıdır. Bu tez kapsamında bu problemlerde göz önüne alınarak oryantasyon sensörlü kafa takibi sistemi ile operatörün kafa hareketlerini takip edebilecek servo motor kontrollü pan-tilt-roll mekanizmasına sabitlenmiş, 3 boyutlu çekim yapabilen video kameradan kişisel üç boyut görüntüleyiciye ve/veya 6 adet LED monitörden oluşan görüntüleme sistemine görüntü aktarımı anlatılmaktadır. Çalışmanın sonunda yapılan testlerle görüntüleme sisteminde 2 boyutlu ve 3 boyutlu görüntünün kullanıcı üzerinde etkileri incelenmiş ve monitöre 3 boyutlu görüntü aktarımda kullanıcının daha rahat hareket edebildiği görülmüştür. Bir diğer testte ise 2 boyutlu görüntünün aktarımı ile kullanıcın kişisel görüntüleyici kullanımı ve 6 ekranlı görüntüleme sistemi kullanımı arasındaki performans farklılığına bakılmıştır. Bu testte de kullanıcın 6 monitörlü sistemi kullanmasının performası üzerinde biraz daha olumlu etki yaptığı görülmüştür.

Anahtar Kelimeler: Oryantasyon sensörü, kafa takibi, servo motor, pan-tilt-roll mekanizması, 3 boyutlu görüntü, video kamera, kişisel üç boyut görüntüleyicisi, LED monitor, görüntüleme sistemi.

To my lovely and sweet family

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere appreciation to my supervisors Assoc. Prof. Dr. E. İlhan Konukseven and Asst. Prof. Dr. A. Buğra Koku for their invaluable guidance, advice, and criticism and especially their extreme support that made this study possible.

I would like to express my thanks to my dear friends and colleagues Akif Hacınecipoğlu, Burak Şamil Özden, Göker Ertunç, Bolkar Altuntaş and all Milvus family for their friendship and technical support throughout the thesis period. I would also like to thank my friends and collegues Serkan Tarçın, Buğra Özütemiz, Matin Ghaziani, Yaser Mohammadi Nazarabad, Kasım Gönüllü, Gökhan Bayar, Kamil Özden, Ozan Yörükoğlu, Ayça İkizer, Dinçer Kaya, and Yılmaz-Esra-Ülkü Ergişi for all the good times we have spent.

I also want to thank my beautiful wife and my love Hatice Duygu Ölmez for her understanding, help, encouragement, faith, love, and for her delicious handmade cooks which all made life easier for me and made this work be completed.

I want to thank to my friendly dad Fazıl Ölmez my lovely mom Hatice Ölmez and my beautiful sister Hilal Ölmez for their support and also for their guidance. I have learned a lot from them. I am grateful to my lovely family for their support, love and encouragement through all my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

SYMBOLS	
$u(t)$	Controller output
K_p	Proportional gain
K_d	Derivative gain
$e(t)$	Error
K_i	Integral gain
τ	Integration variable
Ψ	Yaw Angle
ϕ	Roll Angle

LIST OF ABBREVIATIONS

ABBREVIATIONS

HRI	Human Computer Interaction
AR	Augmented Reality
VR	Virtual Reality
3D	Three Dimensional
HMD	Head Mounted Display
UGV	Unmanned Ground Vehicle
2D	Two Dimensional
USA	United States of America
DIVA	Distributed Interactive Video Array
PTZ	Pan Tilt Zoom
DC	Direct Control
CeDAR	Cable Drive Active vision Robot
IMU	Inertial Measurement Unit
GPU	Graphical Processing Unit
OpenCV	Open-source Computer Vision Library
FPS	Frames per Second
GB	Gigabytes
RAM	Random Access Memory
METU	Middle East Technical University
UPS	Uninterrupted Power Supply

DOF	Degree of Freedom
PDA	Personal Digital Assistant
CMOS	Complementary Metal Oxide Semiconductor
USB	Universal Serial Bus
HDMI	High-Definition Multimedia Interface
DVI	Digital Visual Interface
GPS	Global Positioning System
OpenCL	Open Computing Language
SDK	Software Development Kit
CPU	Central Processing Unit
NED	North-East-Down
EKF	Extended Kalman Filter

CHAPTER 1

INTRODUCTION

Robotics technology which has many application areas is one of the most growing fields. This technology is using in national security, entertainment, search and rescue, earth and space exploration, tactics operations, production, health care and personal assistance.

1.1 Robotics Technology

With the middle of the 20th century, robotics technology has become one of the most active areas of research and began to be used in manufacturing in early 70s. With the 70's robotics started to be used in production and assembly lines. The main use of robotics technology in manufacturing was to replace the repetitive work done by human to decrease accidental risks and the costs with an increase in the production quality. At 2012, over the one million robots are used in industrial applications.

For the past decades, mobile robotics, which involves different scientific disciplines such as computer science, mechanical engineering, electrical engineering, cognitive science, materials science, has been an active research area due to the need of autonomous mobile activities. Different than industrial robots or industrial manipulators, mobile robots have the capability of moving autonomously. This capability makes them popular in various different real life applications. While some of these applications are easy to solve, some others consist hazardous environments and require precise perception, judgment and actuation. They are usually expected to accomplish these tasks by navigating dynamically in those environments consisting humans, other robots, objects and obstacles.

The examples of these applications can be given as car driving, autonomous flight, cleaning, space exploration, delivery, mine sweeping, aiding and rehabilitation. In each application space, the requirements are different and hence, each application has different problems and requires a variety of solutions to these problems. Although the scientific community extensively researched on the field, suggested a bunch of solutions to the different problems related with the area, and even some commercial applications are available, the research field is still intensely active and attractive for the researchers.

1.2 Human-Robot Interaction (HRI)

Mobile robot applications range from fully human-controlled to autonomous agents. In fully teleoperated systems, human has the full control over the perception and movement of the agent. Human perceives and interprets the sensory information and decides on motor commands. Role of the robot is confined to do whatever the operator asks. In fully autonomous systems, human does not intervene to the progress of the robot. Even the need to operate is decided by the robot itself. Although the peak point is defined as fully autonomous robots, human role always will be persistent in mobile robotic applications. In this aspect a new topic arises: Human-Robot Interaction (HRI). HRI study field emerged from the need to understand the interaction between robotic systems and humans.

There should be a communication channel between human and robotic agent to be able to interact. Type of this interaction is defined by the distance between peers. Depending on the distance, human-robot interaction is separated into two categories [1]:

• *Remote interaction:* Defined as the interaction type when the human and the robot are separated in terms of space and/or time.

• *Proximate interaction:* If two peers are in their line of sight, this type of interaction is defined as proximate interaction.

Proximate interaction requires the robot and human being in the same location or even in same room. Interaction between personal robotic assistants and humans require proximate interaction including physical, social or emotive aspects.

Remote interaction included in mobile robotics. If remote interaction with the robot requires mobility, this category is divided into two sub categories [2]. Interaction with a mobile robot to change its location is referred as “teleoperation”. On the other side, if the mission is to change a location of a remote object, this interaction is named as “telemanipulation”.

1.3 Virtual Reality (VR)

The definition of virtual reality comes, naturally, from the definitions for both ‘virtual’ and ‘reality’. The definition of ‘virtual’ is near and reality is what we experience as human beings. So the term ‘virtual reality’ basically means ‘near-reality’. This could, of course, mean anything but it usually refers to a specific type of reality emulation.

Virtual reality is the term used to describe a three-dimensional, computer generated environment which can be explored and interacted by a person. That person becomes part of this virtual world or is immersed within this environment and whilst there, is able to manipulate objects or perform a series of actions.

By using 3D imagery with a head mounted device (HMD) and high quality surrounding sound equipments, these games creates more involvement in the virtual world and consequently shut down the cues of real world. This is what called virtual reality, which has applications far beyond gaming.

On the spectrum between virtual reality, which creates immersive, computer-generated environments, and the real world, augmented reality [3] is closer to the real world. Augmented reality adds graphics, sounds, haptic feedback and smell to the natural world as it exists. Both video games and cell phones are driving the development of augmented reality. Everyone from tourists, to soldiers, to someone looking for the closest subway stop can now benefit from the ability to place computer-generated graphics in their field of vision.

A simple example of ‘Counter Strike’ game can give a thought as to how virtual reality works. The software program for the game is the major element which runs with the help of the computer system and the interfaced input output devices. Every Character and environment within the game behaves closely to reality as per the code written for them. The code facilitates characters and environment to interact with the other characters controlled by the input devices. The code is interpreted by the processor which handles the input – output devices accordingly. This is the simplest example of how VR works. The working of more immersive virtual reality environment is quite similar to working of the game besides the fact that a number of advanced input and output devices along with a high performance processor are added to increase the immersion. The processor executes the processes quickly according to the input given by the user and output is presented to the user in a way that user feels itself a part of the environment and its objects.

1.4 Scope of the Thesis

In this thesis, the primary objective is to design a pan-tilt vision system with a head tracking control. With this system a 3D vision is transferred to a monitor from the camera. Since the pan-tilt mechanism is controlled via head motion, the operator can easily manage the camera where he wants to see.

For this purpose, a pan-tilt-roll mechanism was completed. A 3D video camera is fixed on it. Video camera’s zoom option can be controlled remotely which gives a huge advantage to operator.

Transferred 3D vision is monitoring via 3D eye glass (Head mounted display, HMD) or via a 3D monitor system that includes six 3D monitors.

Pan-tilt mechanism is controlled via head tracking. For tracking, an orientation sensor is used which fixed on a wearable plastic helmet. Since the helmet is wearable, operator can fix the orientation sensor on his head. With operator’s head motion, the position change of his head is calculated and the pan tilt system moves according to this change.

1.5 Motivation of the Thesis

Our Tübitak supported Unmanned Ground Vehicle project, Keçi[1], is planning to control via teleoperation. For this teleoperation process, operator should need a remote control system. Operator should control the Keçi from the base and if it is needed he should take control of the vehicle. For this purpose a vision system is needed. With making this system 3D, the operator should feel like that he is being in the environment where Keçi is located.

After finishing the project mentioned here, the stereo vision system will be transferred on Keçi. So this thesis is one of the basics for our Tübitak supported project.

1.6 Outline of the Thesis

This study is divided mainly two sections according to the content. First section is about development of the pan tilt mechanism that is made ready for vision system. In the second part head tracking is described.

First chapter is the introduction to the topic. Second chapter includes literature survey performed on these two sub-topics. Development of the pan tilt mechanism and head tracking is represented in Chapter 3. In Chapter 4 programming is mentioned. In Chapter 5, the developed head tracking pan-tilt system is validated via experiments. Finally last section is dedicated to conclusions and possible future research on the topic that is studied.

[1] Tübitak Project number 111M580.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

In this chapter a survey on recent research and applications on similar head controlled pan-tilt vision systems are presented. The chapter is divided into two parts. In the first section related work in the literature on pan-tilt mechanism design and manufacture is mentioned. Components, design criterion, sub-systems and application areas are discussed. In the second part of this chapter, head tracking and vision systems are discussed. This part focused mostly on methods on head tracking and on camera systems that are used for teleoperation.

2.2 Literature on Pan-Tilt Mechanism

Pan-tilt systems are used for very wide applications but mostly used for standing a camera. These mechanisms are used in industries, surveillance, medical applications, military, agriculture areas, entertainment applications etc.

Some camera firms such as SONY, Canon and Panasonic produce a commercial pan-tilt camera seen on Figure 2.1. These cameras are widely used in security applications and also in industrial applications.

Figure 2.1- Pan-tilt cameras of SONY [4] and Panasonic [5]

These cameras capture video in 2D. Their prices are very high, from $4000 to $15000 (USA prices). Despite these disadvantages, most academicians use these cameras in their surveys. In a study held by Lang et al. [6] Sony EVID31 pan-tilt color camera is mounted on top of their search robot called BIRON (Figure 2.2). This camera is for acquiring images of the upper body part of humans interacting with the robot. In their research they work on a robot platform which can communicate with people. During interaction the robot gives its attention to the person of interest. The researchers use a multi model person tracking technique for this interaction. They use Sony EVID31 pan-tilt camera for face recognition, two microphones for sound source localization, and a laser range finder for leg detection and called this system as a multi model person tracking. A pan-tilt color camera is used for a tracking of a person of interest.

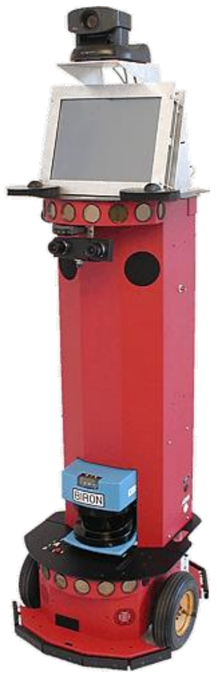


Figure 2.2- BIRON has a Sony EVID31 pan-tilt color camera on the top. [7]

In other research [8], Kogut and his friends use Canon VCC4-R pan-tilt zoom camera for controlling their unmanned ground vehicle (UGV) remotely. They called their project as "The Distributed Interactive Video Array (DIVA)". The aim of this study is to develop a wireless network communicated video system. With this system an operator could easily command UGV remotely. They design man-portable network system with self-calibrating pan-tilt smart cameras for providing usable capabilities to robotic and manned forces.

Another research with using a commercial pan-tilt camera is about Neuroscience [9]. Fry and his friends try to track flying insects with using Sony LSX PT1 pan-tilt camera. They use 2 of this camera for making 3D vision. In their research, they use pan-tilt camera for two different experiments. First they use honey bees for measuring the position and precise body direction while approaching a food source. In the second experiment the flight trajectories of a phonotactic parasitoid fly homing in on its cricket host were recorded in 3D. Pan-tilt cameras are used for following the flying insects' path.

Sinha and Pollefeys use Canon VB-C10 and Sony SNC-RZ30 in their research [10]. In their study, they discuss the problem of recovering the calibration of a network of pan-tilt-zoom cameras. They develop a method for calibrating two different PTZ cameras so that cameras can capture the same video at the same time.

Pan-tilt mechanism cameras can be used in medical applications. Hu et al. [11] develop an insertable surgical pan tilt imaging device. Since this device is a surgical device, they develop very small pan-tilt mechanism provided by two very small DC servo motors with a modular camera. They use this camera system in laparoscopic surgery and the video is transferred to a monitor. The operator makes the operation via looking this monitor. This small device enables to make some surgeries without opening the skin. Hu and his friends develop a remote control for this camera system so that an operator can easily intervene the operation.

Some researchers develop their own pan tilt systems. Troung and his friends design a mechanism for a stereo vision system [12]. They called this system as CeDAR (Cable Drive Active vision Robot). CeDAR is designed for speed and accuracy through its novel use of cables in zero backlash transmissions and a parallel mechanical architecture.

Pan-tilt mechanism can be used in UAVs. Jakobsen and Johnson use pan/tilt/roll mechanism called gamble in their UAV research [13]. In their research they use Georgia Tech's UAV, the GTMax. They develop architecture for controlling the pan/tilt/roll camera system of the GTMax. The camera is fixed on a large gimbal. All of three axes driven by a modified servo motor and the position is checked via optical encoders. They use PID controller for position control of the servo motors.

Another study on UAV is made by Lee et al. [14]. Their research presents a mission-centric approach to controlling the optical axis of a video camera mounted on a camera positioner and fixed to a quadrotor remotely operated vehicle. They control pan/tilt and roll axes of camera fixed gimbal system. They design a camera system and develop architecture for controlling this system.

2.3 Literature on Head Tracking System

In literature, there are lots of techniques for tracking head motion. One of the most common techniques is wearable tracking systems. Foxlin and Harrington study on a system that is based on the very simple idea of combining a head orientation tracker with a head-worn tracking device that tracks a hand-mounted 3D beacon relative to the head [15]. They fix an orientation sensor to HMD (Head Mounted Display). They wear a hand-mounted wireless acoustic emitter beacon for tracking the head motion. For demonstration, they use a portable VR (virtual reality) system and a wearable computer user interface.

Another study for wearable head tracking sensor is done by Joffrion [16]. He uses a low cost MEMS IMU and a low cost single frequency GPS receiver so that the system costs low. The system was flight tested on board a Raytheon C-12C aircraft. The accuracy of the system was measured by comparing its output to truth data from a high-accuracy post-processed navigation grade INS/DGPS solution. Results showed that roll and pitch error were accurate to within 1-2 degrees, but that heading error was dependent upon the flight trajectory.

Foxlin and Naimark [17] are the other researchers who studied on wearable head tracking sensor. They use InertiaCube2 IMU (inertial measurement unit) and fix it on a cap as seen from Figure 2.3. This IMU contains 3 gyros, 3 accelerometers, and 3 magnetometers in a 1 cubic inch package. The operator wears this head-mounted sensor assembly to his head and fixes the belt-mounted electronics unit that houses the CPU board and image processor board to his belt.

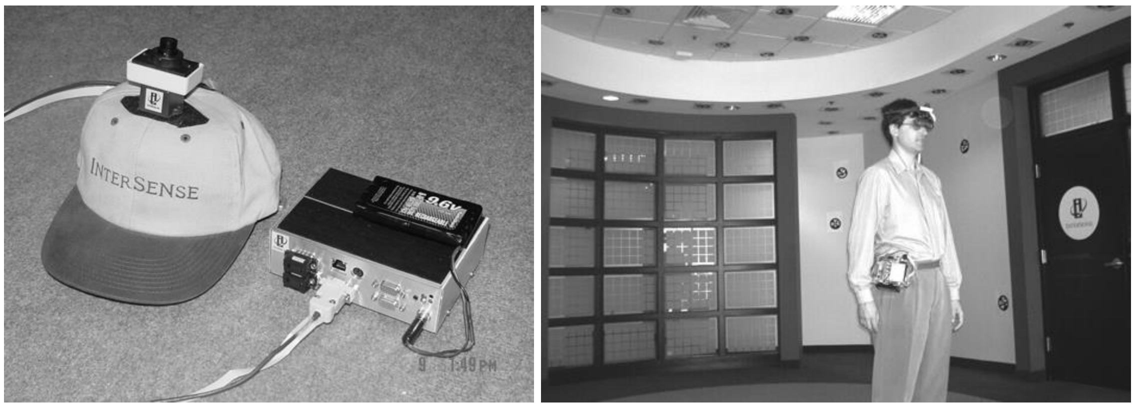


Figure 2.3- Head mounted senor [17]

Another study for the visualization subsystem is made with a laptop with 3D graphics acceleration feeds 3D graphics to an optical-see-through stereo HMD (Figure 2.4) [18]. The laptop handles also interaction with a PDA (via bluetooth) and tracking of interaction devices (using the head-mounted webcam). The tracking subsystem is included a dedicated single board computer which receives input from a camera mounted on top of the helmet (Figure 2.4) from a 6 DoF inertial tracker mounted on the rear of the helmet. It calculates the head pose in real-time and communicates pose information via local LAN to the host (the visualization subsystem). Using the CMOS camera with USB II and an inertial tracker with serial interface, the tracking subsystem is quite compact and can be applied standalone for many kinds of mobile tracking applications.

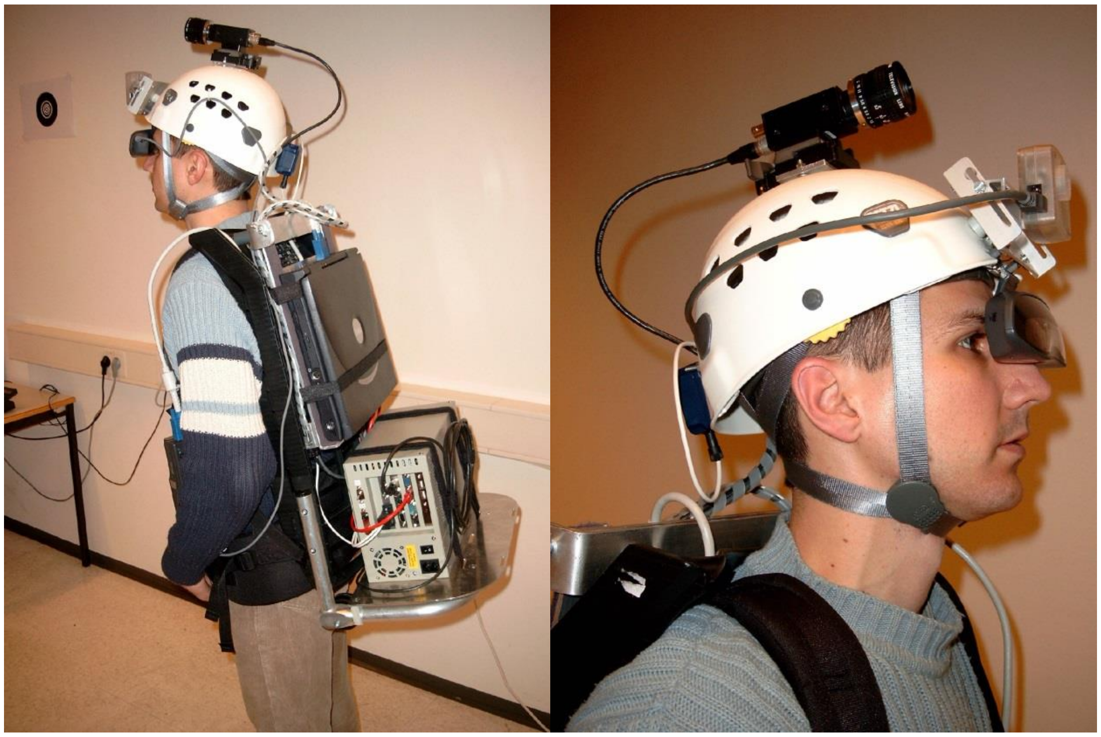


Figure 2.4- Wearable outdoor tracking systems [18].

Foxlin [19] examined the use of use of inertial sensors for head tracking. His system was based on three orthogonal solid-state rate gyros, a two-axis fluid inclinometer and a two-axis fluxgate compass. Orientation was determined by integrating angular rates from the gyros starting from a known initial orientation. Drift compensation was accomplished by using the inclinometer and compass as a “noisy and sloshy but drift-free" measurement of orientation. Estimates of orientation were then generated using a Kalman filter and both sources of orientation. Foxlin implemented an adaptive algorithm by increasing the estimate of inclinometer measurement noise during periods of slosh. On the other hand, the estimate of measurement noise was decreased at a specified length of time since the last nonzero gyro reading or last change in the inclinometer reading. In this way, the Kalman filter took advantage of the inclinometer and compass measurements when they were the most accurate (with no head motion). This technique would not be advantageous in an aviation environment, because several phases of flight, including takeoff and coordinated turns, are exposed to sustained constant linear acceleration.

Foxlin et al. [20] design an inertial system for tracking head mounted displays (HMD)s. They use InertiaCube for inertial measurement. The InertiaCube simultaneously measures 9 physical properties, namely angular rates, linear accelerations, and magnetic field components along all 3 axes. They use SoniDisc for remote-triggered acoustic pulse transmitter. The SoniDisc is a battery powered wireless transponder which receives infrared (IR) signals and transmits ultrasonic pulses in response. They design two different tracking systems: 3 DOF system and 6 DOF system. 3 DOF orientation tracking system is efficient for games, visualizations, and vehicle simulators. 6 DOF tracking system is achieved by adding an ultrasonic range measurement system on 3 DOF system.

Foxlin [21] make another head tracking research which the operator’s platform does not stable at this time. He studies on tracking head relative to a moving vehicle or simulator platform using differential inertial sensors. In this research an inertial measurement unit (IMU) is fixed on a HMD while another IMU (called as reference) is fixed on a platform. This reference IMU’s position relative to ground is known. He derives the kinematic equations and obtain simulation results but concludes that the conclusions can be applied to hybrid inertial trackers involving optical, magnetic, or RF drift correction as well.

Head tracking can be achieved via other techniques such as with intensity gradient and color histogram, with face detection, with eye tracking etc. Xu and Li [22] and Birchfield [23] study on head tracking with camera. At these two researches a method of using particle filter with intensity gradient and color histogram is used for tracking head. Xu and Li model the head’s projection onto the image plane first in their research. Head modeled as an ellipse whose position and size are continually updated by a local search combining the output of a module concentrating on the intensity gradient around the ellipse’s perimeter with that of another module focusing on the color histogram of the ellipse’s interior. Since these two modules have roughly orthogonal failure modes, they serve to complement one another. The result is a robust, real-time system that is able to track a person’s head with enough accuracy to automatically control the camera’s pan, tilt, and zoom in order to keep the person centered in the field of view at a desired size. Extensive experimentation shows the algorithm’s robustness with respect to full 360-degree out-of-plane rotation, up to 90-degree tilting, severe but brief occlusion, arbitrary camera movement, and multiple moving people in the background.

Wang and Brandstein [24] study on a hybrid system based on both voice recognition and face detection for real time head tracking. They use microphone array for detecting the position of speaking person and a stereo camera for face detection. After detecting voice and face, with computer controlled stereo camera they track the face real time regardless of operators motion. The aim of this research is directly video conferences. They use Canon VC-C1 PTZ camera so that they can achieve pan and tilt motion during the tracking.

Hennessey et al. [25] study on an eye tracking system. They track eye-gaze with a single high resolution camera. The camera does not have any moving system so it is a disadvantage for this research when the operator quits the camera's sensor. They achieve free head motion with using 3D models and multiple glints. Over a field of view of 14x12x20 cm, and various configurations such as camera resolutions and frame rates they achieve accuracies under 1° of visual angle.

To sum up, from the literature survey it can be seen that there are various ways of developing pan-tilt mechanisms and tracking head motion. For pan-tilt vision systems researchers generally use commercial products such as pan tilt zoom video cameras. This method is the easiest way to achieve pan tilt vision system. For head tracking, the most common way is to use an IMU for head orientation. The research proposed in this study deals with own designed pan tilt vision system and head tracking with orientation sensor.

CHAPTER 3

DEVELOPMENT OF A HEAD TRACKING PAN-TILT MECHANISM

3.1 Introduction

In this chapter the design of pan-tilt mechanism and head tracking is mentioned. Used equipment is introduced. First a pan-tilt-roll mechanism is mentioned. This part is divided into two subtopics: Mechanical system and vision system. In each sub system requirements, equipment and their properties and technical information is mentioned. After finishing pan-tilt-roll mechanism, head tracking system is mentioned. The sensors, processors are mentioned. Detailed software is mentioned in next chapter.

3.2 Pan-Tilt-Roll Mechanism

Pan-tilt-roll mechanism is composed of two sub systems; mechanical system and the vision system.

3.2.1 Mechanical System

This system includes pan tilt and roll systems which consists of pan-tilt mechanism, roll mechanism, and motors.

3.2.1.1 Pan Tilt and Roll Mechanism

For tracking head motion, pan tilt and roll mechanisms are needed since neck motion can be achieved by combination of these axes. Instead of designing a new mechanism, providing one of the commercial mechanical pan tilt roll system is suitable. So two disassembled Servo City's pan tilt mechanisms are obtained and after combining one of these pan tilt mechanism, the other's tilt mechanism is used as a roll mechanism. The mechanism can be seen on Figure 3.1.

Mechanism consists of two independent axes; pan and tilt. For each axis a standard servo motor is needed. Housings of motor are made with ball bearings which protect motors' shafts. Two axes are tightened via ball bearing supported pan mount. Pan motor's shaft passes through this mount and tightens to tilt mechanism. Servo City comments to use standard analog servo motors such as Hitec HS-485HB or HS-645MG or Futuba BLS151 Analog servo motors. After making some experiments with Hitec HS-485HB, it is seen that this motors do not satisfy our requirements since these recommended motors are analog and have low torque values. As a result digital and high torque Hitec HS-5955TG and Savöx SC-1256 servos are used in this mechanism. Details of the motors are mentioned later.

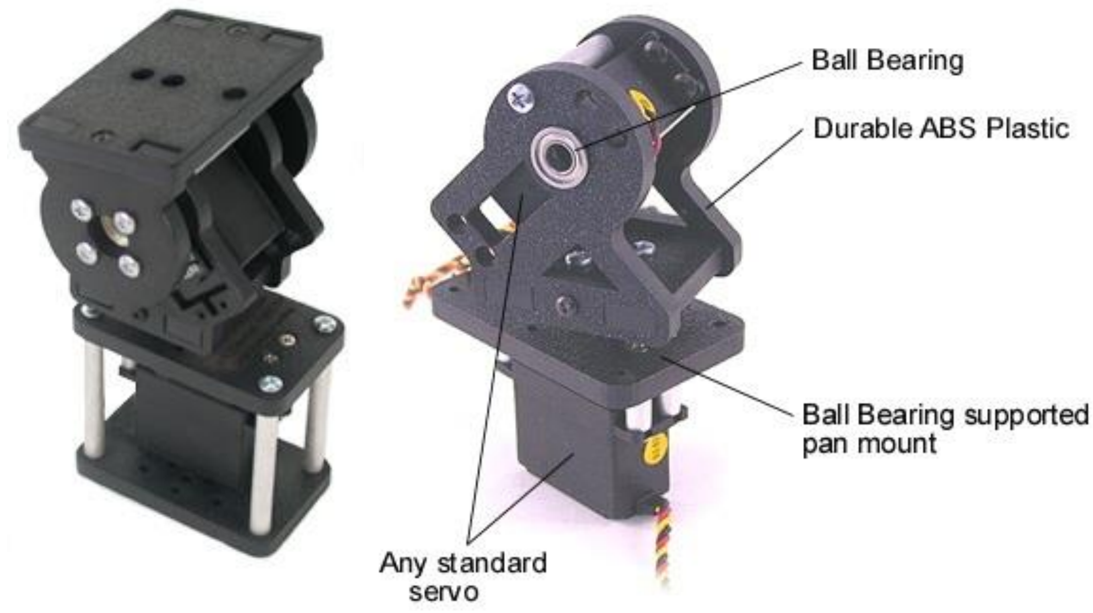


Figure 3.1- Servo City’s pan-tilt mechanism [26]

After combining pan-tilt system, roll axis is added to this system (Figure 3.2). This axis is achieved by adding a tilt mechanism to pan tilt system with tightened to pan mechanism. Roll axis also has ball bearing housing.

On top of the tilt mechanism, the plate designed as a mounting plate so it has holes on it. With using this property, 3D stereo camera is fixed on this plate easily.

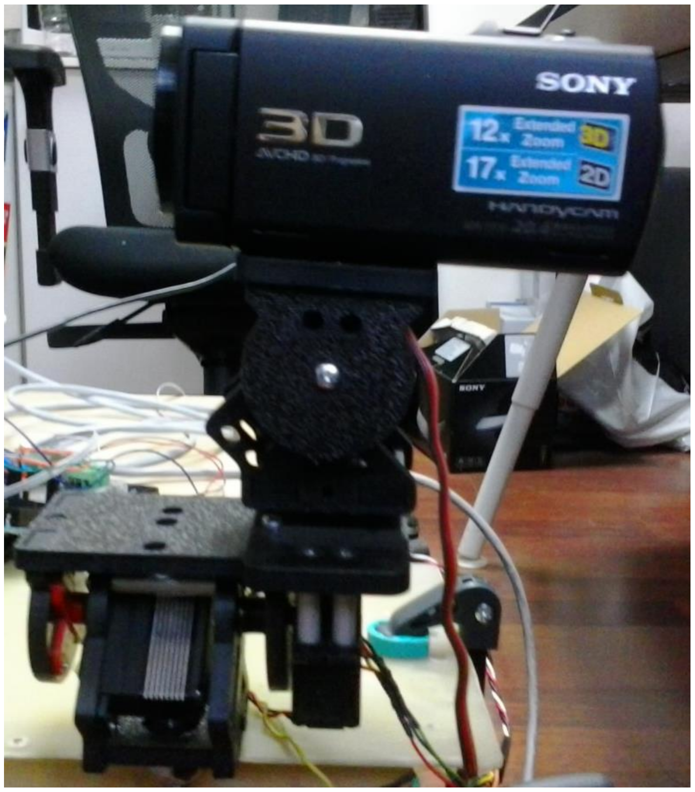


Figure 3.2 - Pan Tilt Roll Mechanism

3.2.1.2 Servo Motors

For roll and pan axis, two Hitec HS-5955TG is used and for tilt axis one Savöx SC-1256 digital servo motor is used.

Up to just a few years ago, the only RC servo motors available were analog, but now digital servo motors are available. There is no physical or main component difference between a digital servo motor and analog servo motor. The servo motor case, motor, gears, and even the feedback potentiometer all have the same functions and operations in both types. The difference between the two is in how the signal from the receiver is processed and how this information is used to send power to the servo motor.

An analog servo motor controls the speed of the motor by applying on and off voltage signals or pulses to the motor. This voltage is constant especially 4.8 to 6.0 volts. This on off frequency is standardized to 50 cycles a second. The longer each on pulse is, the faster the motor turns and the more torque it produces. Most of the motors' speed is controlled with this way. At rest, no current is going to the motor. If a small transmitter command is given or some external pressure is applied to the servo horn forcing it off neutral, a short duration voltage pulse will be sent to the motor. The larger the stick movement or potentiometer movement, the longer this on pulse will be in order to move the servo quickly to the desired position.

During small amounts of stick movement or when external forces are applied forcing the servo off its neutral or holding position; only a short duration voltage pulse is applied to the servo motor every 20 milliseconds. With large stick movements, a long voltage pulse is applied every 20 milliseconds to the servo motor. A short power pulse every 20 milliseconds doesn't get the motor turning that quickly or allow it enough time to produce much torque. This is the problem with all analog servos; they don't react fast or produce much torque when given small movement commands or when external forces are trying to push them off their holding position. This area of slow sluggish response and torque is called dead band.

A digital servo has all the same parts as an analog servo, even the three wire plug that plugs into the receiver is the same. The difference is in how the pulsed signals are sent to the servo motor.

A small microprocessor inside the servo analyzes the receiver signals and processes these into high frequency voltage pulses to the servo motor. Instead of 50 pulses per second, the motor will now receive up to 300 pulses per second. The pulses will be shorter in length of course, but with so many voltage pulses occurring, the motor will speed up much quicker and provide constant torque. Digital servos have a shrill sound when very light force loads placed on them. This sound is the short high frequency voltage pulses acting on the motor.

The result is a servo that has a much smaller deadband, faster response, quicker and smoother acceleration, and better holding power. Digital servos have advantages of increase in speed, torque, and holding power does come with a small disadvantage; power consumption.

To sum up; the analog servo motor is slow to respond and provides little torque during small, fast command inputs. So, digital servo motors are much better than analog servo motors.

A standard 3-pole wire wound servo motor uses a steel core with wires wound around the core, this core is then surrounded by permanent magnets. The core and all that wire weighs a fair bit. However in a Coreless design, the heavy steel core is eliminated by using a wire mesh that spins around the outside of the magnets. This design is much lighter hence it has a smaller inertia which results quicker acceleration and deceleration. The result is smoother operation, more available torque, and faster response time.

So according to information given above, the used servo motors are chosen as a Coreless Digital Servo Motor.

Hitec HS-5955TG is an ultra-torque titanium gear coreless digital servo motor (Figure 3.3). It has a 24 kg.cm torque in 6.0 V DC. It is only 61.5 g with the size 39.9x20.1x38.1 mm (LxWxH). It has a 20 ms pulse cycle with 900-2100 μs duty cycle.

Figure 3.3- Hitec HS-5955TG ultra-torque titanium gear coreless digital servo motor [27]

The other motor is Savöx SC-1256 which is also an ultra-torque titanium gear coreless digital servo motor (Figure 3.4). It has a 20 kg.cm torque in 6.0 V DC. It is a bit lighter than Hitec, 52.4 g. Its size is 40.3x20.2x36 mm (LxWxH). It has a 20 ms pulse cycle with 800-2200 μs duty cycle. It can be controlled via pulse with modulation.

Figure 3.4- Savöx SC-1256 ultra-torque titanium gear coreless digital servo motor [28]

Since Hitec servo motor has larger torque value than Savöx, the Savöx is used as a pan motor while Hitec motors are used as a tilt and roll motors.

3.2.2 Vision System

In this thesis 3D vision is transferred to the operator’s monitor so that the operator can feel that he is in the environment where the video system is being. 3D vision system is the one of the basics of this project.

3.2.2.1 3D Camera

Since 3D vision system has a huge importance for this project, in the design period several alternatives are considered. In the first design, two small 2D cameras with adjustable position according to operator’s eye position are considered. Cams can be positioned for every operator, according to their pupil’s position. With these 2 small cams, 2 visions are transferred in real time. With processing these two visions a 3D vision is generated and is transferred to operator’s monitor. However, transferring these 2 videos synchronously cause problems despite the small cams are the same.

After failed first design, Sony 3D camera “HDR-TD30V” is considered (Figure 3.5). It is a 3D handycam that enable to transfer 3D vision in real time. After making tests with HDR, it is seen that the specifications of camera is fairly enough for this project. So HDR is used as a visual system 3D camera.

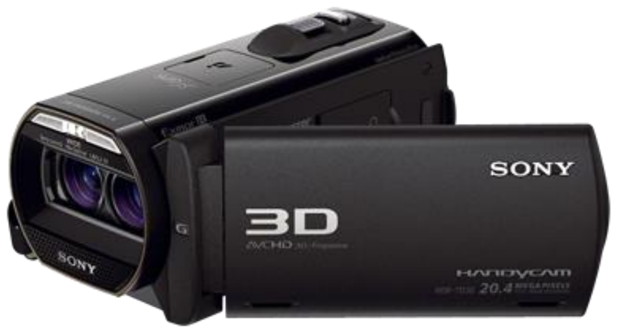


Figure 3.5-Sony HDR-TD30V [29]

Sony HDR-TD30V has two Full HD sensors as seen from the Figure 3.5. It can capture videos 1920x1080 resolution on both sensors. Each sensor independently capture 1920x1080 resolution video which results as more clear high definition (HD) 3D videos. Unlike Sony HDR most 3D cameras achieve the 3D affect by splitting a 1920x1080 signal in to two 960x540 sensors in order to squeeze them onto one sensor, results limiting resolution on the final output.

Some other important features of the HDR-TD30V are listed as below:

- Size (LxHxW) : 63.5 x 71.5 x 131 mm,
- Weight : 460 g (without battery),
- HDMI output,
- USB output (as a mass storage),
- Optical Zoom : 10x,
- Digital Zoom : 120x (2D only),
- Lens Stabilization : Optical Steady Shot image stabilization with active mode (Wide 3D),
- Focal Length (35mm equivalent) : 33.4mm-400.8mm (16:9 Movie Mode, 3D),
- Minimum Focus Distance : Approximately 30cm (Wide 3D),
- Focal Distance : f = 3.2-32mm(3D),
- Built in GPS,
- Remote control.

3D video is transferred via HDMI output. Video can be transferred in real time while the camera records. It has 10x optical zoom and it can be controlled remotely like play and stop. With using remote control property, the operator can remotely control zoom, stop and play.

3.2.2.2 Decoding Remote Control of 3D Camera

For controlling the camera’s zoom, stop and play properties remotely, remote codes of the camera is needed. For this purpose a set-up is made with two controller board, Arduino MEGA R3 and Arduino UNO R3. One controller is for receiving infrared codes from the remote controller and the other is for testing; transferring the codes to the camera. (Figure 3.6)

Receiver board is connected to the computer so that the remote codes can be seen from the serial monitor. 38 kHz SFH 506 is used for receiving codes from the remote control (Figure 3.7). With pushing the buttons of the remote control, a remote code of this pushing button is received and seen on the monitor. After applying this method to all buttons of remote control, all of the infrared codes are obtained. The receiving code of Arduino MEGA R3 is given in Listing 1.

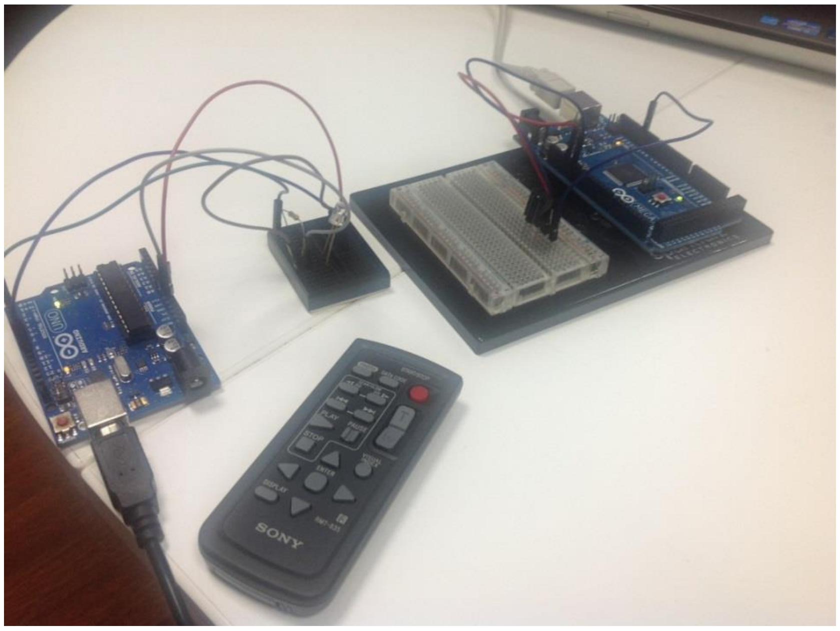


Figure 3.6-Obtaining remote codes of HDR-TD30V

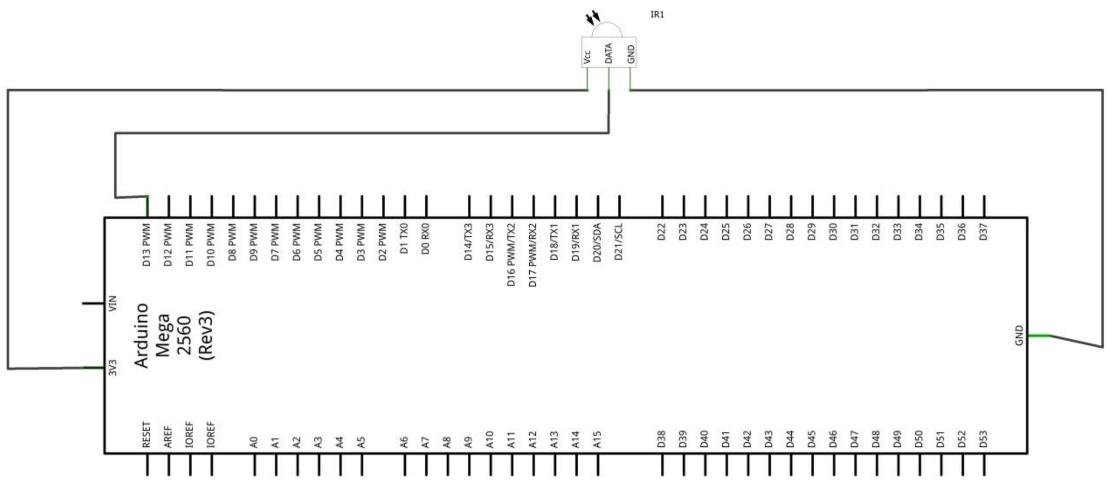


Figure 3.7- Infrared receiver board

```
#include <IRremote.h>
int RECV_PIN = 11
IRrecv irrecv(RECV_PIN)
decode_results results
void setup
  Serial.begin(9600)
  irrecv.enableIRIn
  void dump(decode_results *results)
  int count = results->rawlen
  if (results->decode_type == SONY)
    Serial.print("Decoded SONY: ")
  else if
    Serial.print("Decoded UNKNOWN: ")
  Serial.print(results->value, HEX)
  Serial.print(" (")
  Serial.print(results->bits, DEC)
  Serial.println(" bits)")
  Serial.print("Raw (")
  Serial.print(count, DEC)
  Serial.print("): ")
  for (int i = 0; i < count; i++)
    if ((i % 2) == 1)
      Serial.print(results->rawbuf[i]*USECPERTICK, DEC)
    else
      Serial.print(-(int)results->rawbuf[i]*USECPERTICK, DEC)
        Serial.print(" ")
    Serial.println("")
void loop()
  if (irrecv.decode(&results))
    Serial.println(results.value, HEX)
    dump(&results)
    irrecv.resume()
```

Listing 1- Receiving remote codes of HDR-TD30V

After receiving the remote codes of HDR-TD30V, it is checked whether the codes are working or not. For this purpose a transmitter board with infrared led is connected to Arduino UNO R3 (Figure 3.8). The most useful parameters' codes, start/stop, zoom in, zoom out, are tested.

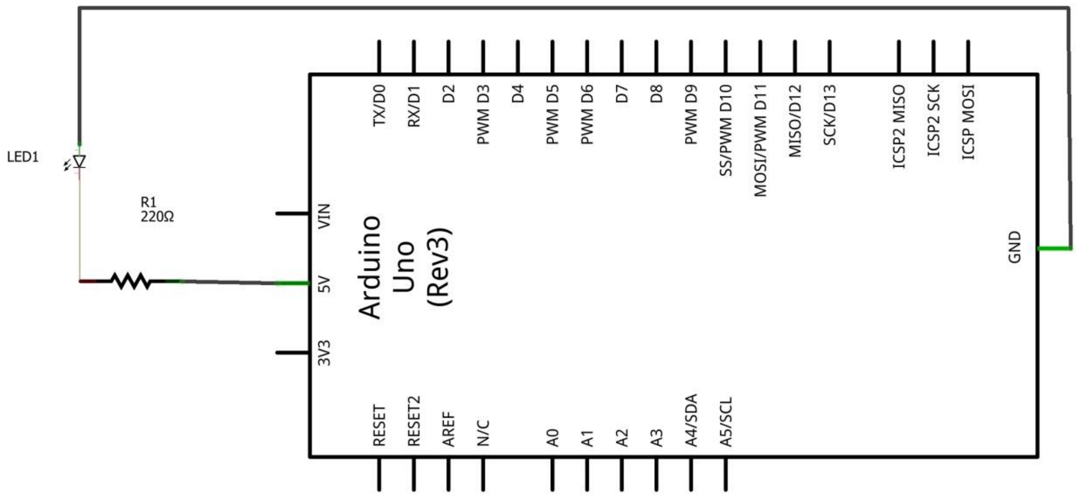


Figure 3.8- Infrared transmitter board

For activating the remote code, code should send 3 times via infrared led. The example transmitting code of Arduino UNO R3 for zoom in is given in Listing 2.

```
#include <IRremote.h>
IRsend irsend
unsigned int zoomIn[31] = {2400, 550, 650, 550, 1200, 550, 650, 550, 1200, 550, 1250,
550, 600, 550, 650, 550, 1200, 600, 600, 550, 650, 550, 1200, 550, 1250, 550, 600, 550,
1250, 550, 1200}
void setup()
void loop()
   irsend.sendRaw(zoomIn,31,38)
   delay(50)
   irsend.sendRaw(zoomIn,31,38)
   delay(50)
   irsend.sendRaw(zoomIn,31,38)
   delay(2000)
```

Listing 2- Transmitting remote codes of HDR-TD30V

Although remote codes of all buttons are decoded, zoom in (T), zoom out (W) and start/stop codes are used in this project. These codes are listed in Table-1.

A joystick system shown in Figure 3.9 is adapted to system and operator can easily control the zoom property of HDR-TD30V by pushing the joystick forward and backward. With adding BC547 - NPN BJT transistor to this remote control board, the range of infrared control is increased.

Table 1- Some of the important remote codes

Name of the Button	**Remote Code of Button**	**Raw Data (Hex)**
Start/ Stop	CBC0	32118 2400 -550 650 -550 600 -600 600 -550 650 -550 1200 -550 1200 -600 600 -550 650 -550 1200 -550 650 -550 1200 -550 1250 -550 1200 -550 1250 -550 600 -550 650 -550 600 -600 600 -550 650 -550 600
Zoom in (T)	2C9B	-17046 2450 -500 650 -550 1250 -500 650 -550 1250 -550 1200 -550 650 -550 600 -550 1250 -550 650 -500 650 -550 1250 -500 1250 -550 650 -500 1250 -550 1250
Zoom out (W)	6C9B	-23238 2400 -600 1200 -500 1250 -600 600 -550 1200 -550 1250 -550 600 -600 600 -550 1250 -550 600 -550 650 -550 1200 -600 1200 -500 650 -600 1200 -550 1200

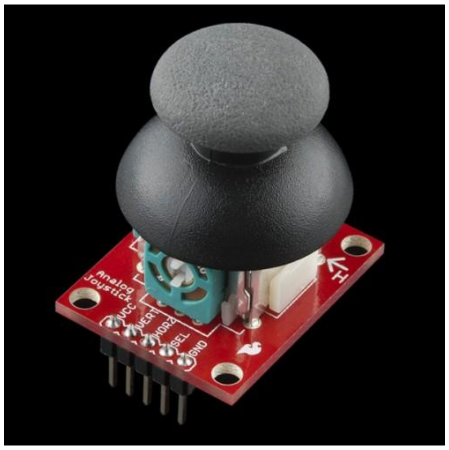


Figure 3.1-Joystick for controlling zoom in and out of HDR-TD30V

3.2.2.3 3D monitor

After transferring 3D vision from the Sony HDR-TD30V, the operator needs a 3D display. For this purpose two display systems are available; a wearable 3D viewer SONY HMZ-T2 (Figure 3.10), and a display system with combining six 24 inch ASUS VG248QE led monitors (Figure 3.11).

Sony HMZ-T2 is a wearable 3D viewer. It has a processor unit and HMZ-T2 should be connected to this unit. The processor unit converts 2D vision to 3D and also converts 3D visions to proper format for showing on the HMZ-T2.

Its weight is 330 grams. It has twin OLED screens with 1280x720 resolutions. Its size is 210x196x110 mm. It has a 3.5 cable length from the unit processor. Its power consumption is 15W. It has a highly adjustable headband and forehead support for a secure and gentle fit.

Operator should use this wearable display for virtual reality. Since he sees on the displays where the camera captures the video, he feels like he is on the camera’s environment. The video is transferred to HMZ-T2 via its processor unit.

The other alternative for 3D display system is combining six of 24 inch ASUS VG248QE led monitors (Figure 3.11). When the operator wants to see the video in detailed and large scale he should use this system.

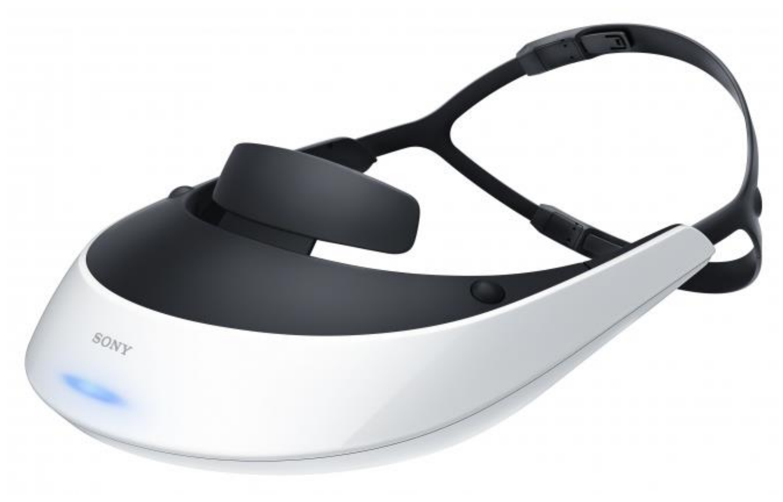


Figure 3.2-SONY HMZ-T2 [30]

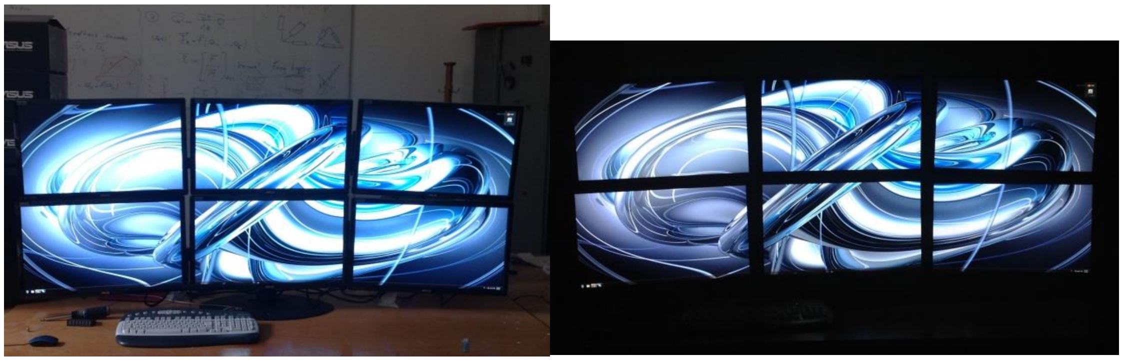


Figure 3.3-Six monitor display system

For this system six ASUS VG248QE led monitors are combined. Six 24 inch LED monitor desk mount stand is ordered from the manufacturer. Three of these monitors are fixed on the upper side and the rest is on the lower side of the stand so that 2x3 24 inch LED monitor display system is prepared.

Captured video is distributed via a graphics card ASUS HD7970 DIRECTCU II (Figure 3.12) to this six display system. This graphics card enables to show a captured video at six displays, 2 of them is via DVI input and 4 of them is via HDMI input. It is an AMD Radeon card and it has a 3 GB RAM. The monitors can be connected to this card via DVI and Display Port. It has 5600 MHz memory speed with a core speed 1000MHz. Its memory is GDDR 5 type with a 384 bits interface. It supports DX11.1.

Figure 3.4- ASUS HD7970 DIRECTCU II [31]

A video capture card (Figure 3.13) is used for transferring video from the HDR-TD30V to computer which ASUS HD7970 is fixed. This card is suitable for HDMI input and for High Definition videos.

Figure 3.5-Avertv HD High Definition Video Capture Card [32]

3.3 Head Tracking Design

Head tracking system consists of one orientation sensor fixed plastic helmet and one controller board Arduino MEGA. Orientation sensor is mounted at a polystyrene helmet so that operator can easily put the sensor on his head. (Figure 3.14)

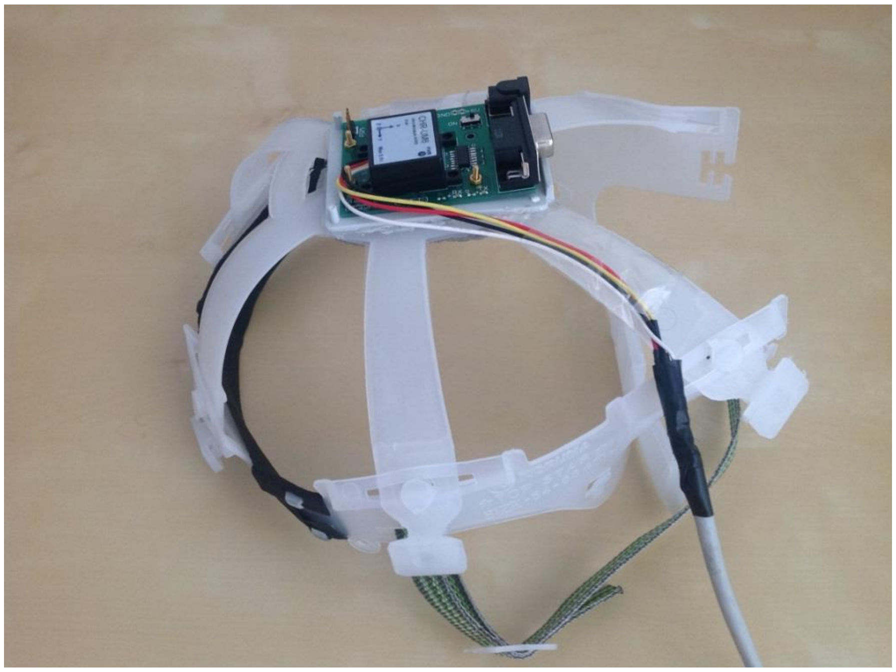


Figure 3.6- Polystyrene helmet for fixing orientation sensor

3.3.1 Orientation Sensor

An orientation sensor is used for determining the position of operator’s head. For this purpose, “UM6 Orientation Sensor” is used (Figure 3.15). The UM6 is designed by CH Robotics. It is a miniature sensor and has a protect case with the sizes 28.2x27.2mm and a height of 11.5mm.

The UM6 has 3-axis accelerometer, angular rate sensor, and compass sensor modules. UM6 measures orientation in all three dimensions at 500 Hz using a combination of rate gyros, accelerometers, and magnetic sensors. By combining data from all three types of sensors, the UM6 produces orientation measurements that are resistant to vibration and immune to long-term angular drift.

The UM6 provides orientation measurements using Euler Angles (yaw, pitch, and roll) and quaternions. In addition to orientation, the UM6 reports raw and processed sensor data from the accelerometers, rate gyros, and magnetic sensors. The UM6 also has the capability to interface with external GPS modules to provide position, velocity, course, and speed information.

The sensor included an onboard 32-bit ARM processor which fused the data from the individual sensor modules using an Extended Kalman Filter algorithm and provided the processed data using serial transmission at a rate of up to 1 KHz. With the automatic gyro bias calibration and the cross-axis misalignment correction, the sensor could be easily calibrated using free open source software to correct for sensor biases, allowing for very accurate orientation data.

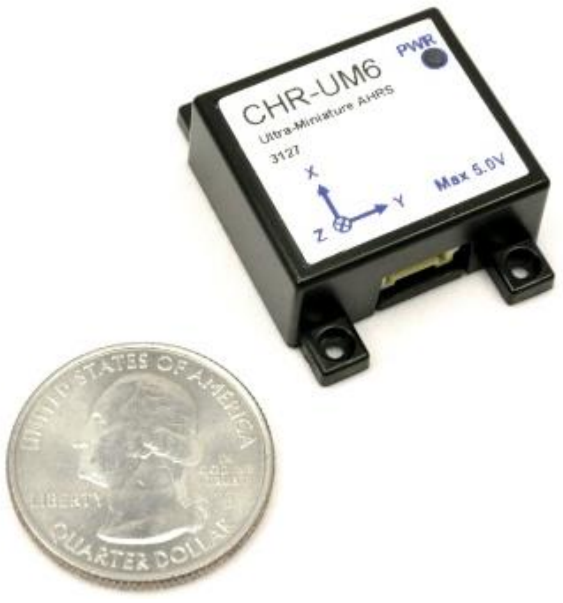


Figure 3.7- The UM6 Orientation Sensor [33]

From the tests, it is seen that the UM6 has accuracy better than 2^o for pitch and roll angle and better than 5^o for yaw angle. It can measure +/- 2000 °/s rotation rates and +/- 2g acceleration. It needs 5V input voltage for working.

UM6 supports angle estimation using Euler Angles or quaternions, in this project as mentioned later Euler Angles outputs are used. In Euler Angle mode, for yaw angle the UM6 restricts magnetometer updates. As a result in environments where the external magnetic field varies significantly it is important to keep accurate pitch and roll angle estimates.

The UM6 has own serial interface software to simplify configuration and testing, and can be connected to PC via serial connection. For making serial connection easily, a serial connection board, also developed by CH Robotics, is obtained (Figure 3.16). On a serial connection board, there is a socket for fixing the UM6, a voltage jack and a serial connection jack.

By using CHR Serial Interface software, sensor settings can be changed, real time data logging can be achieved and sensor calibration can be done (Figure 3.17).

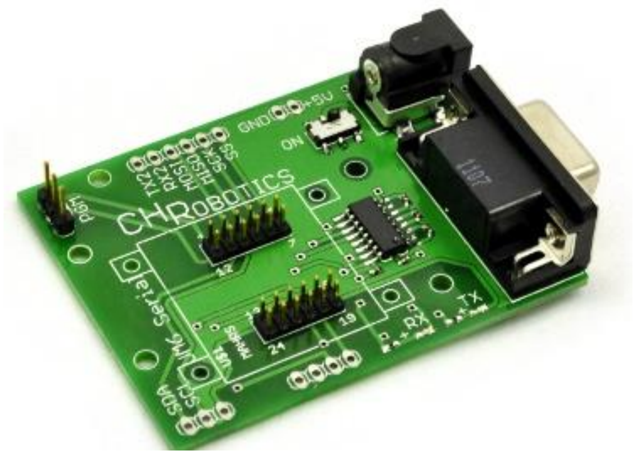
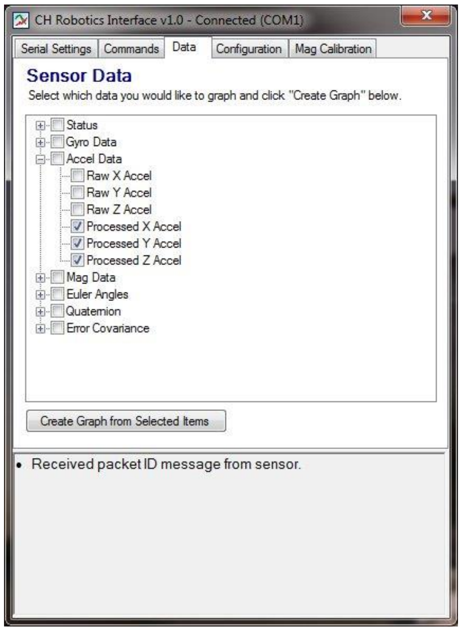


Figure 3.8-UM6 serial connection board

The UM6 can provide orientation information using both Euler Angles and Quaternions. Compared to quaternions, Euler Angles are simple and intuitive and they lend themselves well to simple analysis and control. On the other hand, Euler Angles are limited by a phenomenon called "Gimbal Lock". In this thesis Euler Angles are used instead of Quaternions since in applications where the sensor will never operate near pitch angles of +/- 90 degrees, Euler Angles are still a good choice.


CH Robotics Interface v1.0 - Connected (COM1)

Serial Settings | Commands | Data | Configuration | Mag Calibration

Sensor Data

Select which data you would like to graph and click "Create Graph" below.

- Status
- Gyro Data
- Accel Data
  - Raw X Accel
  - Raw Y Accel
  - Raw Z Accel
  - ✓ Processed X Accel
  - ✓ Processed Y Accel
  - ✓ Processed Z Accel
- Mag Data
- Euler Angles
- Quaternion
- Error Covariance

Create Graph from Selected Items

- Received packet ID message from sensor.

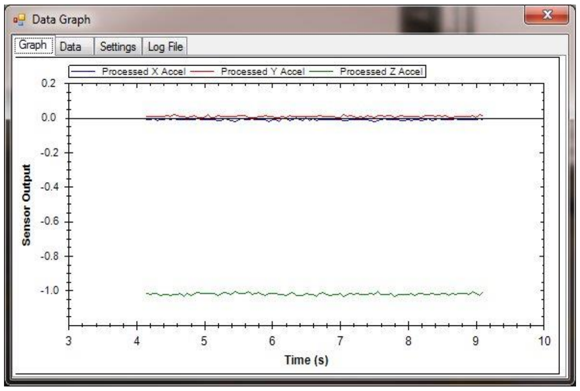


Figure 3.9- CHR Serial Interface software

3.3.1.1 Euler Angles

Euler angles provide a way to represent the 3D orientation of an object using a combination of three rotations about different axes [34]. Multiple coordinate frames are used to describe the orientation of the sensor, including the "inertial frame," the "head-1 frame," the "head-2 frame," and the "body frame." The inertial frame axes are Earth-fixed, and the body frame axes are aligned with the sensor. The head-1 and head-2 are intermediary frames used for convenience when illustrating the sequence of operations that take us from the inertial frame to the body frame of the sensor [35].

The "inertial frame" is an Earth-fixed set of axes that is used as an unmoving reference. Common aeronautical inertial frame are used where the x-axis points north, the y-axis points east, and the z-axis points down (Figure 3.18). It is called as a North-East-Down (NED) reference frame. Since the z-axis points down, altitude above ground is actually a negative quantity.

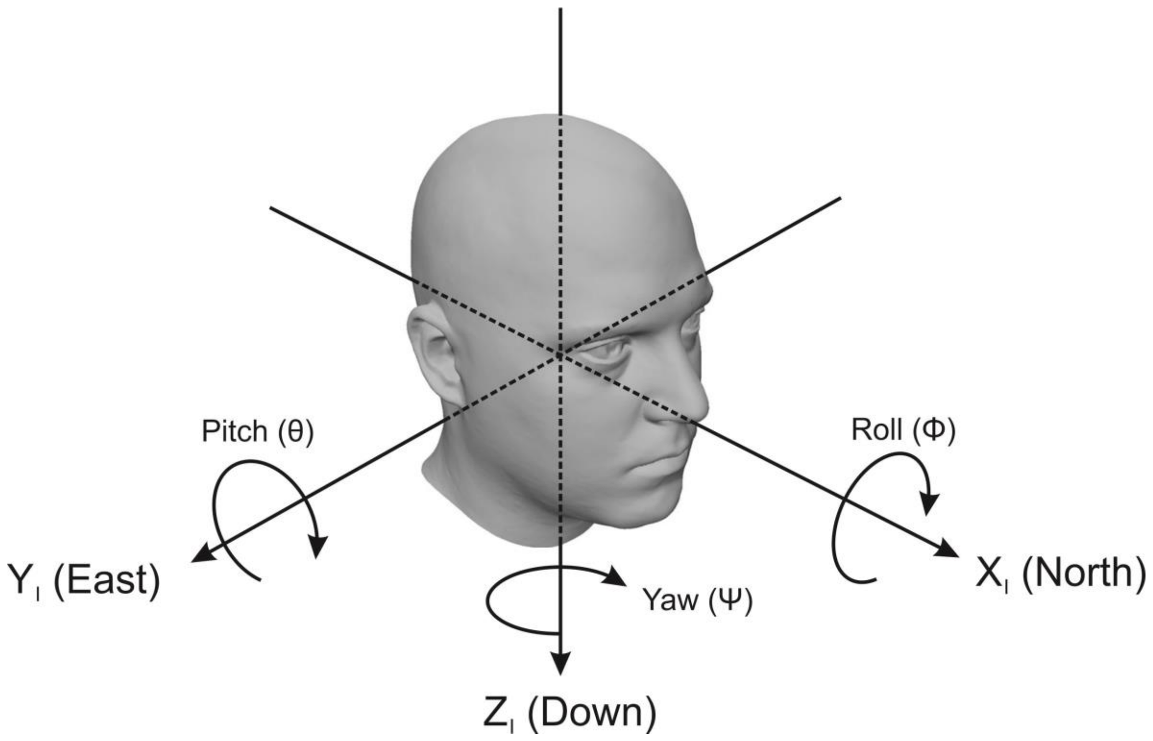


Figure 3.10- Inertial Frame: Roll-Pitch-Yaw Angles

The sequence of rotations used to represent a given orientation is first yaw, then pitch, and finally roll. The "pitch" angle represents positive rotation about the y-axis, "roll" represents positive rotation about the x-axis, and "yaw" represents positive rotation about the z-axis.

Yaw angle represents rotation about the inertial-frame z-axis by an angle Ψ (Figure 3.18). The yaw rotation produces a new coordinate frame where the z-axis is aligned with the inertial frame and the x and y axes are rotated by the yaw angle Ψ. This new coordinate frame is called as the head-1 frame. The orientation of the head-1 frame after yaw rotation is shown in Figure 3.19. The head-1 frame axes are colored red, while the inertial frame axes are gray.

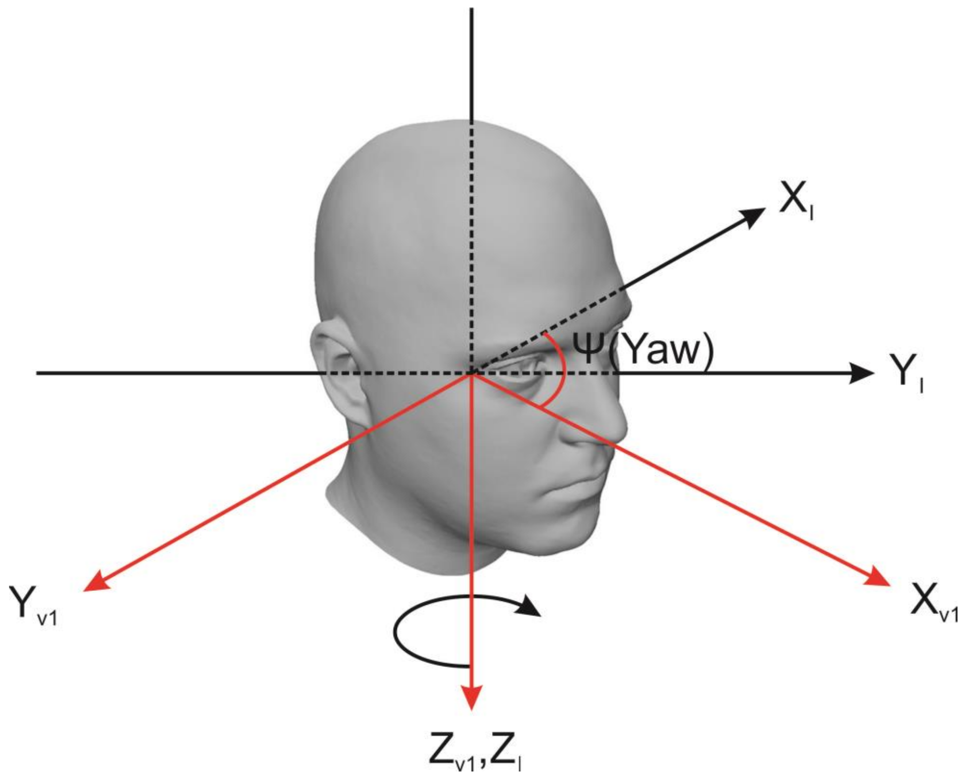


Figure 3.11- The Head-1 Frame

Rotation of a vector from the Inertial Frame to the Head-1 Frame can be performed by multiplying the vector by the rotation matrix.

$$R_I^{\,v1}(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Pitch represents rotation about the head-1 Y-axis by an angle θ as shown in Figure 3.20. For clarity, the inertial-frame axes are not shown. The head-1 frame axes are shown in gray, and the head-2 axes are shown in red. Assume that pitch isn't rotation about the inertial-frame Y-axis.

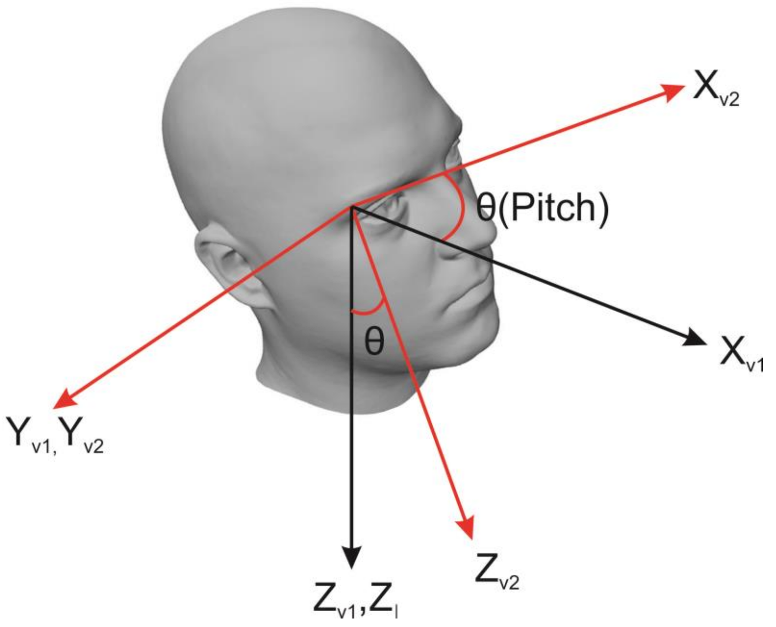


Figure 3.12-The Head-2 Frame

The rotation matrix for moving from the head-1 frame to the head-2 frame is given by

$$R_I^{v2}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

The rotation matrix for moving from the inertial frame to the head-2 frame consists simply of the yaw matrix multiplied by the pitch matrix:

$$R_I^{v2}(\theta,\psi) = R_{v1}^{v2}(\theta) R_I^{v1}(\psi).$$

The body frame is the coordinate system that is aligned with the body of the sensor. On an aircraft, the body frame x-axis typically points out the nose, the y-axis points out the right side of the fuselage, and the z-axis points out the bottom of the fuselage.

The body frame is obtained by performing a rotation by the angle ϕ around the head-2 frame x-axis as shown in Figure 3.21. For clarity, the body frame and head-1 frame axes are not shown. The head-2 frame axes are shown in gray, while the body-frame axes are shown in red.

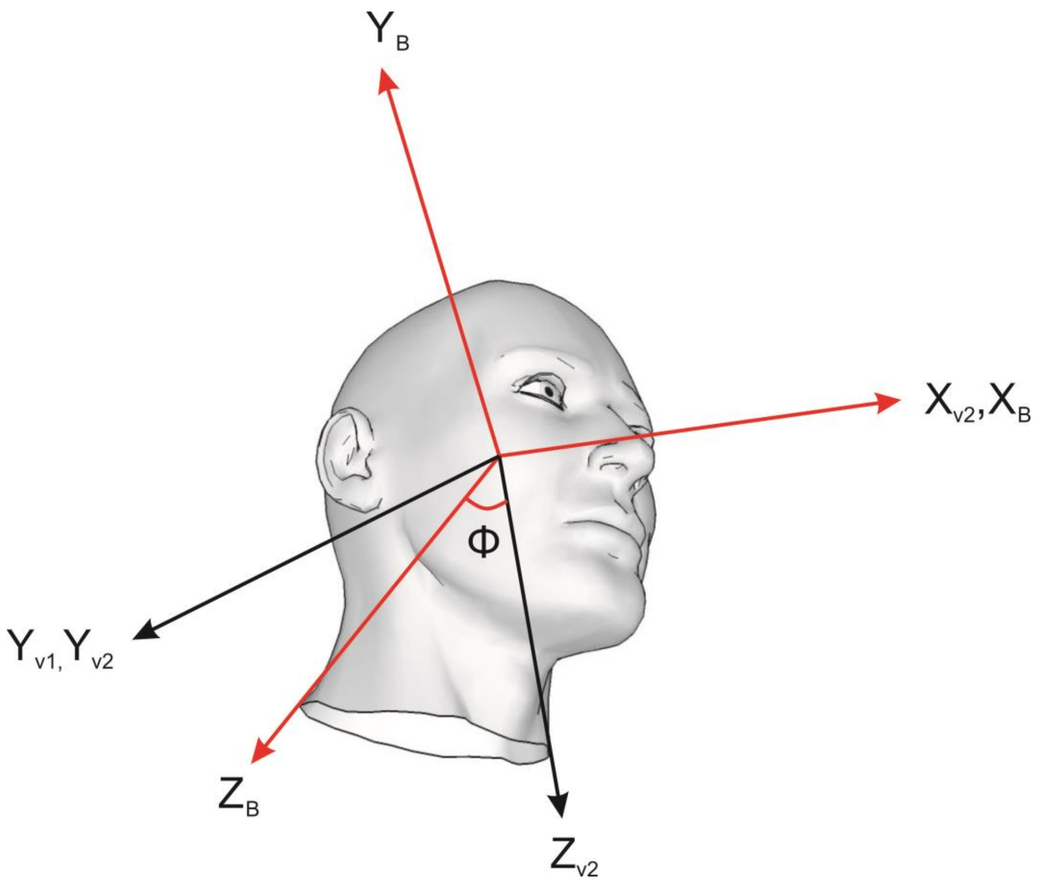


Figure 3.1-The Body Frame

The rotation matrix for moving from the head-2 frame to the body frame is given by

$$R_{v2}{}^{B}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}$$

The complete rotation matrix for moving from the inertial frame to the body frame is given by

$$R_{I}{}^{B}(\phi,\theta,\psi) = R_{v2}{}^{B}(\phi) R_{v1}{}^{v2}(\theta) R_{I}{}^{v1}(\psi).$$

The rotation matrix for moving the opposite direction - from the body frame to the inertial frame - is given by

$$R_{B}{}^{I}(\phi,\theta,\psi) = R_{I}{}^{v1}(-\psi) R_{v2}{}^{B}(-\theta)(-\phi) R_{v1}{}^{v2}.$$

3.3.1.2 Transformation of Euler Angles to Camera System

Axes of rotation of head in pan tilt roll motions can be assumed to intersect at one origin at the neck of a human [36]. Therefore angle of motion transformation is not required at head.

Since head motion axes intersect at neck, operator’s eye and head fixed UM6 rotate at the same rigid body and same rotation axis. As a result the angle change of eye and head fixed orientation sensor is same which means no transformation is required.

The rotation axes of pan tilt roll mechanism of 3D camera are shown in Figure 3.22. Roll axis and pan axis of the mechanism intersect at one point as shown from the figure. With taking this point as an origin of the pan-tilt roll motion, tilt axis should be transformed to origin for distance change of camera.

The difference between pan-tilt roll camera mechanism and head neck motion is the translation in the tilt motion.The translation distance (d) is 9.5 cm which is shown on Figure 3.23. This transformation distance (d_T) can be calculated with;

$$d_T = 2d\sin(\frac{\theta}{2})$$

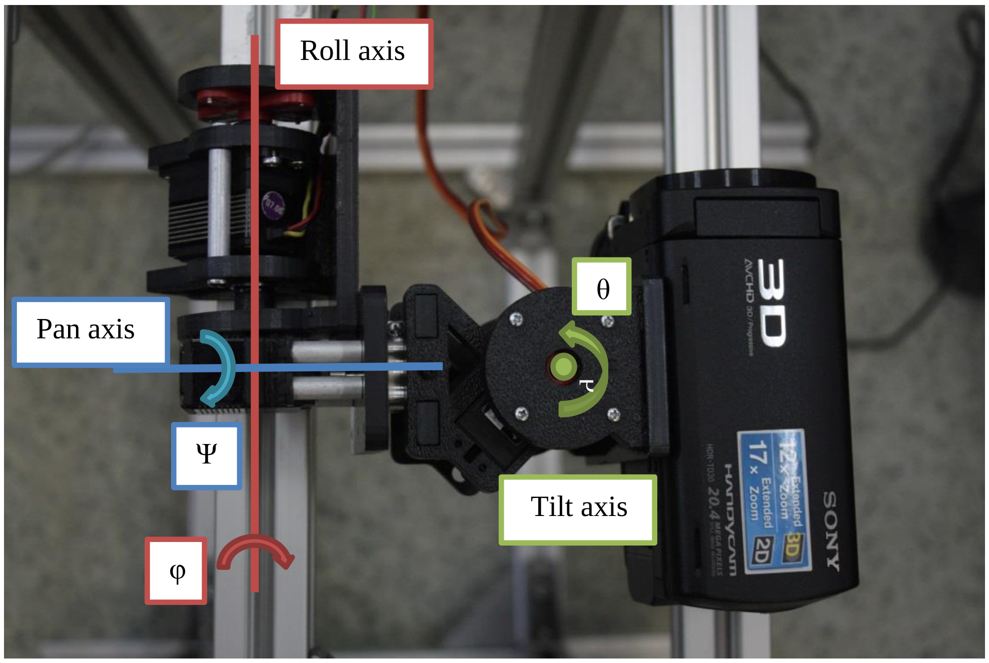


Figure 3.2- The rotation axes of pan tilt roll mechanism

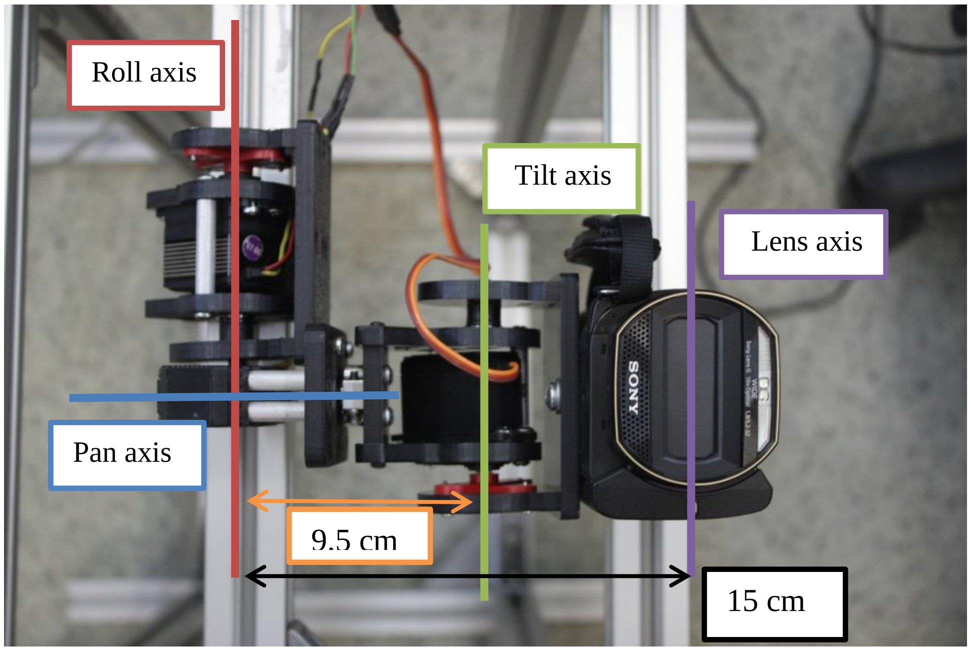


Figure 3.3- Transformation Distances of pan tilt roll mechanism

For 30^0 rotation this transformation distance can be calculated as 4.9 cm for this setup.

As can be seen from the Figure 3.23, the rotation axes are decoupled which makes the transformation of rotation angles redundant. Since there exists no translation in tilt motion, rotation angles should be same to keep axes parallel to each other. The position change of camera causes change of field of view. However, due to presence of Sony's HMD field of view is already smaller compared to field of the necked eye. Therefore this small change of field of view is discarded in this study. Because of these reasons Euler angles obtained from orientation sensor can be directly used as servo rotations angles in the mechanism.

The average value of neck eye distance is 14.7 cm for a man [37]. In the pan tilt roll mechanism the distance of camera's lens to origin is 15 cm as can be seen from Figure 3.23. Therefore motion of camera and the motion of operator's head are nearly same.

3.3.1.3 Gimbal Lock

Gimbal lock occurs when the orientation of the sensor cannot be uniquely represented using Euler Angles. The exact orientation at which gimbal lock occurs depends on the order of rotations used. On UM6, the order of operations results in gimbal lock when the pitch angle is 90 degrees.

Intuitively, the cause of gimbal lock is that when the pitch angle is 90 degrees, yaw and roll cause the sensor to move in exactly the same fashion.

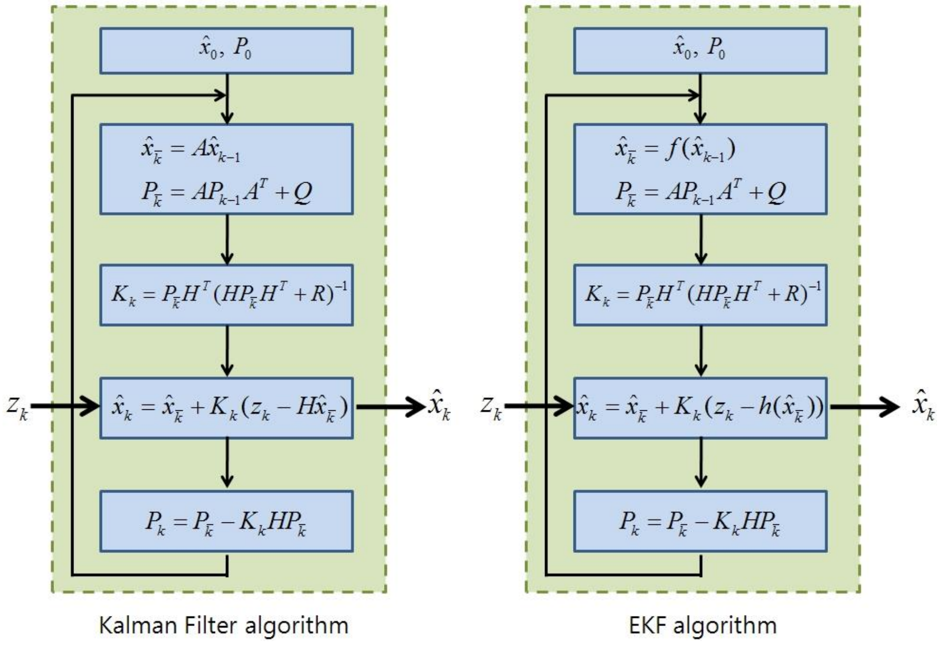
3.3.1.4 Kalman Filter

The Kalman filter was developed by Rudolf E. Kalman. Its purpose is to use measurements observed over time, containing noise and other inaccuracies, and produce values that tend to be closer to the true values of the measurements and their associated calculated values. The Kalman filter has many applications in technology, and is an essential part of space and military technology development. The Kalman filter produces estimates of the true values of measurements and their associated calculated values by predicting a value, estimating the uncertainty of the predicted value, and computing a weighted average of the predicted value and the measured value. The most weight is given to the value with the least uncertainty. The estimates produced by the method tend to be closer to the true values than the original measurements because the weighted average has a better estimated uncertainty than either of the values that went into the weighted average. For the linear system, linear Kalman filter is used and for the non-linear system, extended Kalman filter (EKF) can be applied [38]. Figure 3.22 shows each algorithm.

Z_k, the measurement is the input of the Kalman filter algorithm. And it uses A, H, Q, R from the system model. Then x_k becomes final output. Linear Kalman filter algorithm and extended kalman filter algorithm are almost same. But as the system model cannot be derived from the nonlinear system, linearization of system model is needed for EKF algorithm. f(x_k), h(x_k) are the nonlinear function from the nonlinear system. Jacobian can be used for the linearization.

$$A \equiv \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_k}, H \equiv \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{\bar{k}}}$$

Accelerometer and gyroscope are hard to use separately to measure the position angle. Position angle that derived from the accelerometer has noise, while the gyroscope shows the drift. Though accelerometer gives noisy output, its measuring error is conservative. On the other side, gyroscope gives relatively noise immuned output for short time duration, though its error will be diverged for long time period. With these characteristics, we can see that two sensors are in complementary relationship. If gyro drift can be corrected by accelerometer, the output angle can have reliability. Likewise, by combination of two complementary sensors, improved output can be calculated. This is the concept of the Kalman fusion.



$\hat{x}_0, P_0$
$\hat{x}_{\bar{k}} = A\hat{x}_{k-1}$
$P_{\bar{k}} = AP_{k-1}A^T + Q$
$K_k = P_{\bar{k}}H^T(HP_{\bar{k}}H^T + R)^{-1}$
z_k
$\hat{x}_k = \hat{x}_{\bar{k}} + K_k(z_k - H\hat{x}_{\bar{k}})$
$\hat{x}_k$
$P_k = P_{\bar{k}} - K_kHP_{\bar{k}}$
Kalman Filter algorithm
$\hat{x}_0, P_0$
$\hat{x}_{\bar{k}} = f(\hat{x}_{k-1})$
$P_{\bar{k}} = AP_{k-1}A^T + Q$
$K_k = P_{\bar{k}}H^T(HP_{\bar{k}}H^T + R)^{-1}$
z_k
$\hat{x}_k = \hat{x}_{\bar{k}} + K_k(z_k - h(\hat{x}_{\bar{k}}))$
$\hat{x}_k$
$P_k = P_{\bar{k}} - K_kHP_{\bar{k}}$
EKF algorithm

Figure 3.4- Kalman and Extended Kalman Filter [38]

To get the reliable position angle (Euler angle) with accelerometer and gyroscope, Kalman fusion is needed. Figure 3.23 shows the concept of Kalman fusion in IMU.

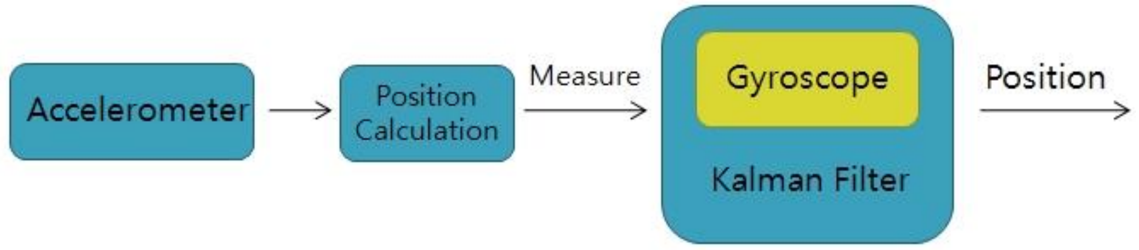


Figure 3.5-Kalman fusion in IMU [38]

To perform the Kalman fusion, system model, A, H, R, and Q should be decided. Interested variables are $\phi, \theta, and\ \varphi$. So, the state variable can be set as like follows:

$$x = \begin{Bmatrix} \phi \\ \theta \\ \varphi \end{Bmatrix}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + w$$

Where w is the noise. From this relationship, the system model becomes;

$$\begin{bmatrix} p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ q\cos\phi - r\sin\phi \\ q\sin\phi\cos\theta + r\cos\phi\cos\theta \end{bmatrix} + w \ = \ f(x) + w$$

Since the system model is nonlinear, it should be linearized by using Jacobian matrix.

$$\begin{bmatrix} p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ q\cos\phi - r\sin\phi \\ q\sin\phi\cos\theta + r\cos\phi\cos\theta \end{bmatrix} \rightarrow A = \begin{bmatrix} \frac{\partial f_1}{\partial \phi} & \frac{\partial f_1}{\partial \theta} & \frac{\partial f_1}{\partial \varphi} \\ \frac{\partial f_2}{\partial \phi} & \frac{\partial f_2}{\partial \theta} & \frac{\partial f_2}{\partial \varphi} \\ \frac{\partial f_3}{\partial \phi} & \frac{\partial f_3}{\partial \theta} & \frac{\partial f_3}{\partial \varphi} \end{bmatrix}$$

So, from the above the system matrix A can be derived. The measured system can be modeled with the noise *v* as;

$$z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} \phi \\ \theta \\ \varphi \end{Bmatrix} + v$$
$$= Hx + v$$

In this case, the system model is linear. So, simply system matrix H can be obtained. Finally, the system matrices A and H are derived. With this system model, Kalman filter algorithm can be applied. CH Robotics implements this algorithm to inside the UM6.

3.3.2 Controller Board

Arduino MEGA R3 is used as a controller board (Figure 3.24). It is an open-source development platform that based on a simple input output (i/o) board. It implements the Processing/Wiring language. It is used for receiving position data of operator’s head from the UM6 and sending this data to motor drivers. Arduino MEGA R3 is used for this purpose since it is easy to programming.

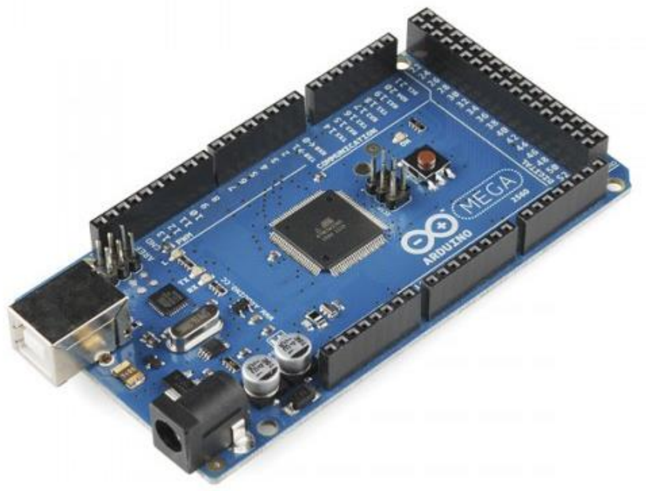


Figure 3.6-Controller Board:Arduino MEGA R3 [39]

It works with 7-12 V. It uses ATmega2560 microcontroller. It has 54 digital inout/output pins, 14 of them is for PWM, 16 analog outputs and 4 UARTs. It can connect to computer via USB meanwhile it can power up via USB. On the board there is an adapter jack. It can also be powered up via this jack. It has 256k flash memory and 16 MHz clock speed.

It can be programmed on own software or programs like Visual Studio.

In Figure 3.25, the connection layout of Arduino Mega is seen. As can be seen one, Arduino's one analog input is used (A0) for joystick (Figure 3.9), communication pins RX1 and TX1 is used for transferring data from UM6, four digital outputs are used, number 2 is for tilt motor, number 4 is for pan motor, number 6 is for roll motor, and number 9 is for remote control board, and one digital input pin, number 5, is used for joystick's push specification. With pushing joystick, pan tilt roll mechanism is return to its starting position, so that operator could initialize servo motors positions after wearing the UM6 fixed plastic hat or when needed. Arduino, motors and UM6 all are powered from 20 A limited 5-12 V voltage regulator. Grounds of all electronic components are mating.

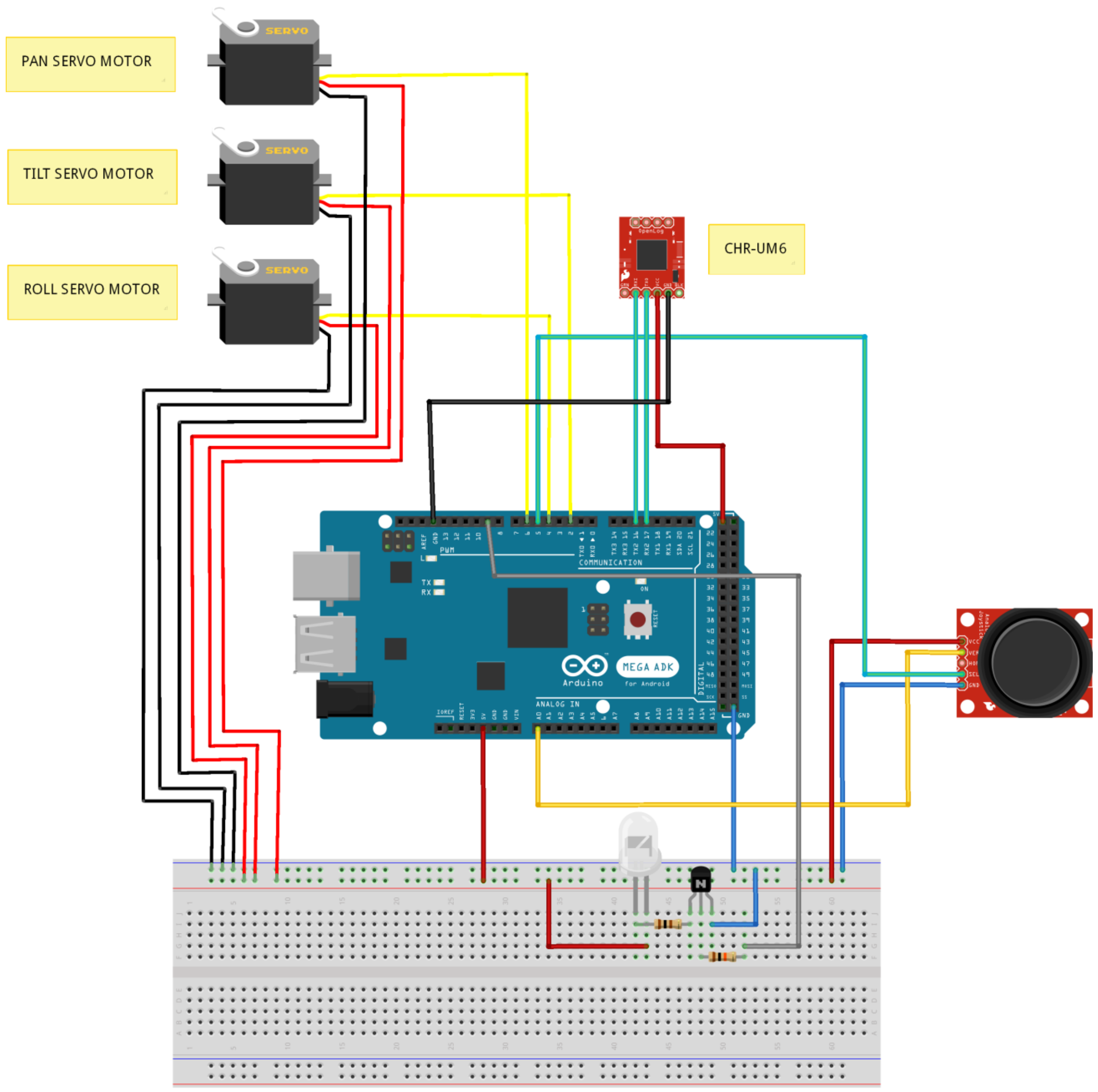


Figure 3.1-Connections of Arduino Mega R3

3.4 Conclusion

In this chapter development of a head tracking pan-tilt mechanism is presented. First pan tilt system is mentioned. At the earlier steps of design the system is considered as a pan tilt system but after seeing the requirement of roll action, roll mechanism is added to system. For virtual reality 3D vision system is needed. At the earlier stages, with combining two small 2D cameras 3D vision is tried to establish. However with the synchronism problem of transferring these two visions, establishing 3D vision via two small 2D cameras is failed. As a result Sony 3D stereo handycam HDR-TD30VE is obtained and adapted to vision system.

HDR-TD30VE is a 3D camera with two lenses. It has image stabilization property which gives a huge advantage for stabilize vision. Some properties of this camera can control via remote control and with decoding the remote codes zoom in and out properties controlled remotely.

For monitoring 3D vision, 2 monitor systems are used; Sony’s HMD, HMZ-T2, and combined 6 ASUS 3D LED monitors system are used.

For head tracking system CH Robotics’ miniature orientation sensor UM6 is used. Sensor is fixed on a plastic helmet for wearing on head and the outputs of Euler angles are used for position change of head. The outputs are interpreted on Arduino Mega R3 and the position changes are transferred to digital coreless RC servo motors. So head tracking is achieved.

In the next chapter, the developed algorithm is introduced.

CHAPTER 4

ALGORITHM STRUCTURE

4.1 Introduction

In this section calibration of orientation sensor and developed algorithm on Arduino Mega R3 is mentioned. This chapter is divided into four parts; calibration of UM6, head tracking algorithm, pan-tilt-roll algorithm and remote control algorithm. Calibration of UM6 is done via CHR Serial Interface software. Algorithm of Arduino is developed on Microsoft Visual Studio 2012.

4.2 Flowchart

Before developing software, a flowchart is drawn which can be seen in Figure 4.1. In this flowchart the steps of how the software works can be seen.

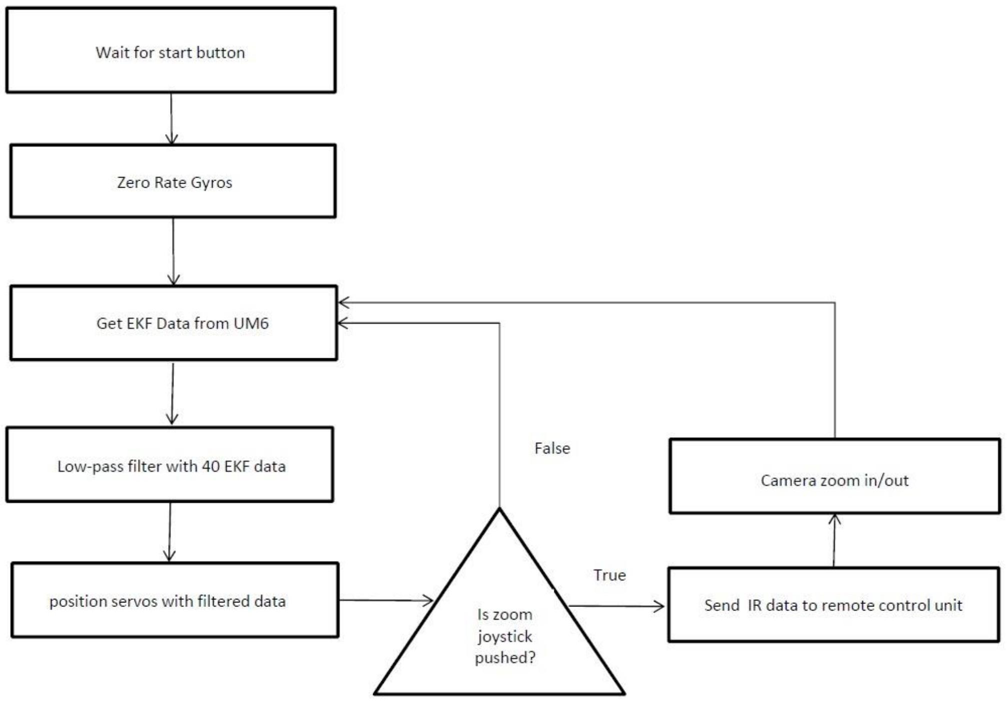


Figure 4.1-Algorithm Flowchart

4.3 Calibrating Orientation Sensor

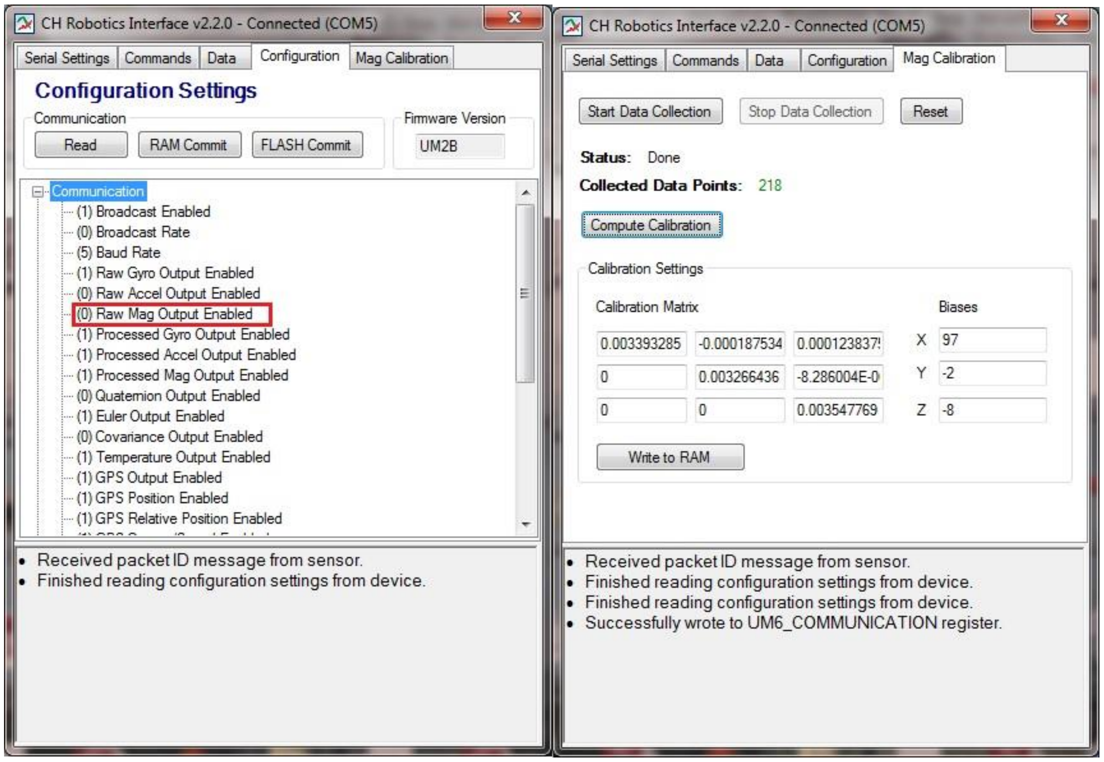
The UM6 is calibrated on factory; however CH Robotics suggests calibrating the sensor on working platform again. There are a lot of ways to calibrate the UM6, but most of them aren't necessary to get reasonable performance from the sensor. The most important requirements are to calibrate the magnetometer and zero the rate gyros.

4.3.1 Magnetometer Calibration

Despite the sensor comes pre-calibrated, CH Robotics suggests after integrating UM6 on its assembly platform where it works, progressing magnetometer calibration make the sensor works fine. Calibration can be performed on CHR Serial Interface software. Before starting calibration, from the configuration tool, raw magnetometer data should turn on (Figure 4.2). After turning on the magnetometer data on, from mag calibration window the calibration can be started by clicking “Start Data Collection” button.

The software increments the "Collected Data Points" indicator as raw data arrives. While collecting the data, for obtaining fine results the sensor should be rotated through a wide variety of yaw, pitch, and roll angles. During the calibration, the sensor should be far from any magnetic field distortions such as computer monitor, ferrous metals etc.

With collecting 300 data points, “Compute Calibration” button enables to click for computing the calibration matrix (Figure 4.2). Finally with clicking “Write to RAM” button, calibration coefficients are sent to the UM6’s RAM. Making magnetometer calibration once is enough and no need to make this progress again.



Configuration Settings

Communication

Firmware Version: UM2B

- Communication
  - (1) Broadcast Enabled
  - (0) Broadcast Rate
  - (5) Baud Rate
  - (1) Raw Gyro Output Enabled
  - (0) Raw Accel Output Enabled
  - (0) Raw Mag Output Enabled
  - (1) Processed Gyro Output Enabled
  - (1) Processed Accel Output Enabled
  - (1) Processed Mag Output Enabled
  - (0) Quaternion Output Enabled
  - (1) Euler Output Enabled
  - (0) Covariance Output Enabled
  - (1) Temperature Output Enabled
  - (1) GPS Output Enabled
  - (1) GPS Position Enabled
  - (1) GPS Relative Position Enabled

- Received packet ID message from sensor.
- Finished reading configuration settings from device.

Status: Done

Collected Data Points: 218

Calibration Settings

Calibration Matrix				Biases
0.003393285	-0.000187534	0.000123837!	X	97
0	0.003266436	-8.286004E-0	Y	-2
0	0	0.003547769	Z	-8

- Received packet ID message from sensor.
- Finished reading configuration settings from device.
- Finished reading configuration settings from device.
- Successfully wrote to UM6_COMMUNICATION register.

Figure 4.2-Configuration Tool and Magnetometer Calibration

4.3.2 Zero the Rate Gyros

Unlike magnetometer calibration, rate gyro biases should be zeroed every time the sensor starts up and periodically while running. The rate gyros are very sensitive to temperature variation, and to keep costs low they are not calibrated in the factory. This means that for best results, they should be zeroed periodically. In figure 4.3 and 4.4 data graph of rate gyros before zeroing and after zeroing is seen. As can be seen rate gyros are not zero after starting the UM6 so that it should be zeroed. The Arduino code for zeroing rate gyros is given in Listing 3. As suggested, the data is zeroed when the sensor starts and periodically during progressing when sensor is stationary.

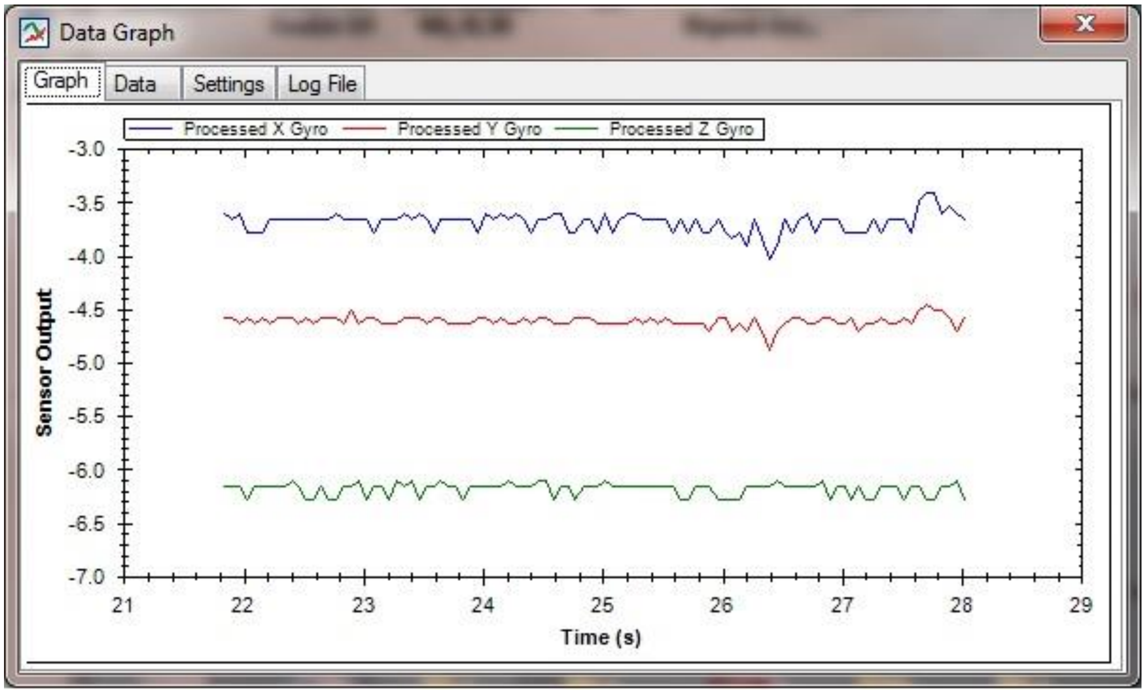


Figure 4.1-Rate Gyro Output Before Zeroing

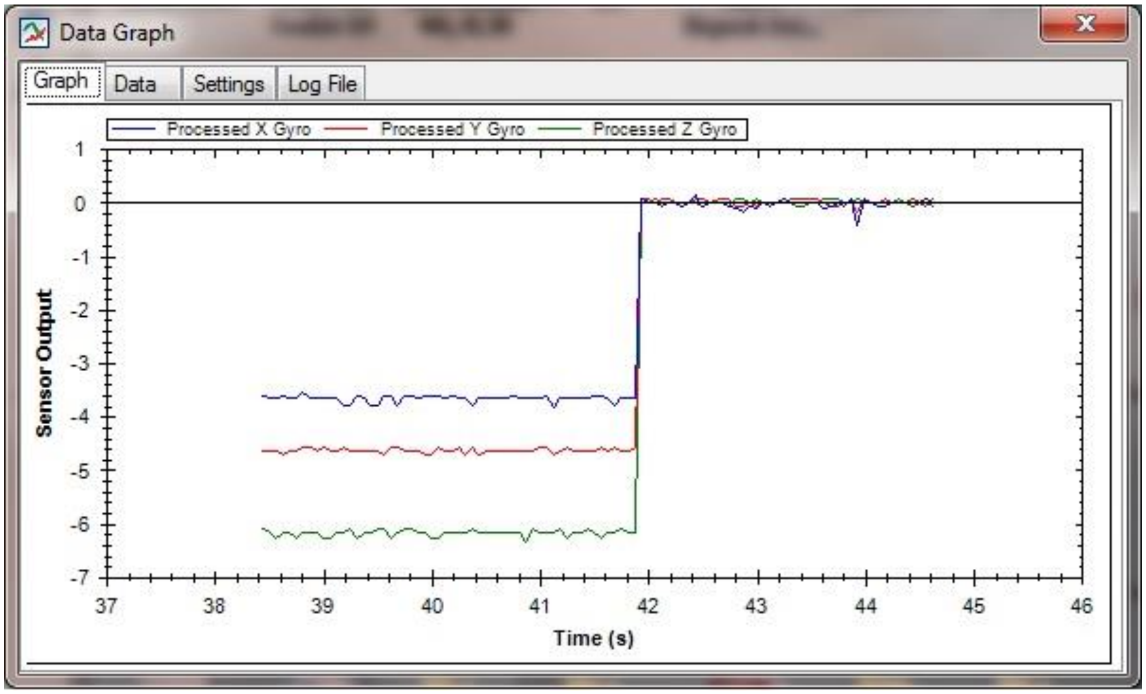


Figure 4.2-Rate Gyro Output After Zeroing

UM6 and Arduino Mega communicate via UART communication with 115200 baud rate, none parity and stop bits of 1.

UM6 supports UART communication for two modes: Broadcast and Listen mode. In Broadcast Mode, any combination of raw or processed sensor data, orientation estimates, and estimator covariance can be transmitted automatically. UM6 does not wait for any request. However in Listen mode, first a request must be sent, then UM6 sends desirable data. In this project Listen mode is used instead of Broadcasting mode.

UM6's UART serial communication structure is given in Table 2.

Table 2- UART Serial Packet Structure

's'	'n'	'p'	packet type (PT)	Address Data Bytes (D0...DN-1)	Checksum 1	Checksum 0

When communication is performed over the UART, data transmitted and received by the UM6 is formatted into packets containing:

1. The three character start sequence 's', 'n', 'p' to indicate the start of a new packet (i.e. start new packet)

2. A "packet type" (PT) byte describing the function and length of the packet

3. An address byte indicating the address of the register or command

4. A sequence of data bytes, the length of which is specified in the PT byte

5. A two-byte checksum for error-detection

The PT byte specifies whether the packet is a read or a write operation, whether it is a batch operation, and the length of the batch operation (when required). The PT byte is also used by the UM6 to respond to commands. The specific meaning of each bit in the PT byte is given in Table 3.

Table 3- Packet Type (PT) Bit Descriptions

Bit(s)	Description
7	Has Data: If the packet contains data, this bit is set (1). If not, this bit is cleared (0).
6	Is Batch: If the packet is a batch operation, this bit is set (1). If not, this bit is cleared (0)
5-2	Batch Length (BL): Four bits specifying the length of the batch operation. Unused if bit 7 is cleared. The maximum batch length is therefore $2^4 = 16$
1	Reserved
0	Command Failed (CF): Used by the UM6 to report when a command has failed. Unused for packet sent to the UM6.

For zeroing the gyro rate, the address byte is given as 0xAC. So with the view of this information, the sent data should be; int RegZeroGyro [7] = {0x73, 0x6E, 0x70, 0x00, 0xAC, 0x01, 0xFD}. The code sending this data to UM6 is given in Listing 3.

```
int RegZeroGyro [7] = {0x73, 0x6E, 0x70, 0x00, 0xAC, 0x01, 0xFD};
        void setup()
        {
           Serial2.begin(115200);
           zeroGyro();
        }
        void zeroGyro()
        {
            int i = 0;
            while(i <= 6)
                  {
                   Serial2.write(RegZeroGyro[i]);
                     i++;
                   }
                  delay(200);
                  Serial2.flush();
        }
```

Listing 3- Arduino code for zeroing rate gyro bias

At the end of the code, “Serial2.flush” is used for waiting the transmission of outgoing serial data to complete.

4.3.3 Calibrating EKF Variables

UM6 orientation sensor has own internal Extended Kalman Filter (EKF) for eliminating noises and irrelevant data. This Kalman Filter has variances that can be calibrated from CH Robotics software interface. These variances are; mag, accelerometer and process variances. From factory UM6 comes pre-calibrated with initial values of 2, 2 and 0.5 with the order of mag, accelerometer and process variances. With these values the sensor has a drift especially on yaw angle as can be seen from Figure 4.5. Sensor is in stationary condition after 11th second however as can be seen from Figure 4.5 it has a drift after that second.

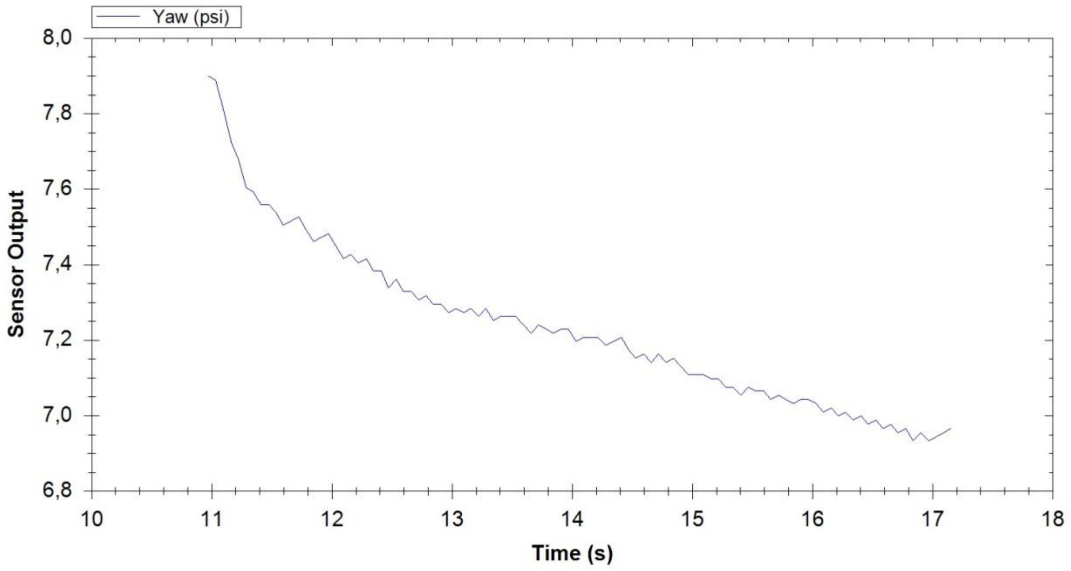


Figure 4.3-Yaw angle drifts with initial EKF variances values.

For eliminating this drift the values of variances are changed. First, lower values are tried, 0.5, 0.5 and 0.1 with the order of mag, accelerometer and process variances. With these values the sensor first overshoots and then continues to drift as can be seen from Figure 4.6. After 168th second, the sensor remains stable, however as can be seen first it overshoots, then starts to drift. So, lower EKF variances values are not a good choice for UM6.

After unsuccessful attempt with lower values of EKF variances, the values are increased. They are given as 10, 50 and 500. With these values the yaw angle starts to oscillate on stationary condition of orientation sensor. So, higher values of EKF variances are not good too for orientation sensor.

After trying some other values, it is seen that choosing values of 2, 5 and 175 for mag, accelerometer and process variances are good for UM6 orientation sensor. With these values sensor has no drift, no overshoot and no oscillation as can be seen from Figure 4.8. So, the calibration of orientation sensor is achieved with the values of mag variance 2, accelerometer variance 5 and process variance 175.

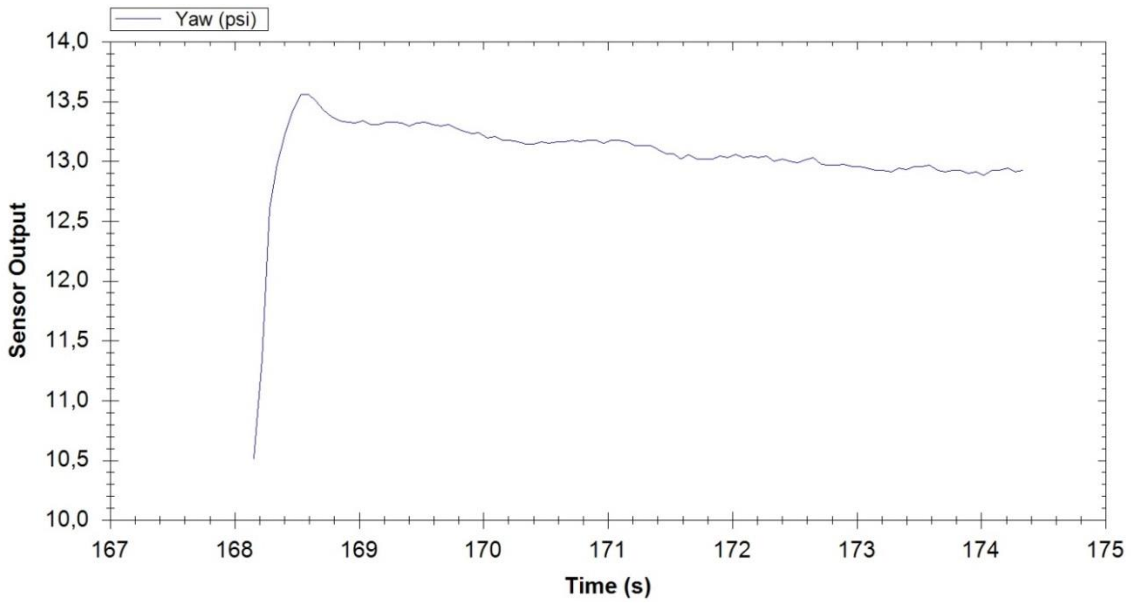


Figure 4.1-Yaw angle first overshoots then drifts with lower EKF variances values.

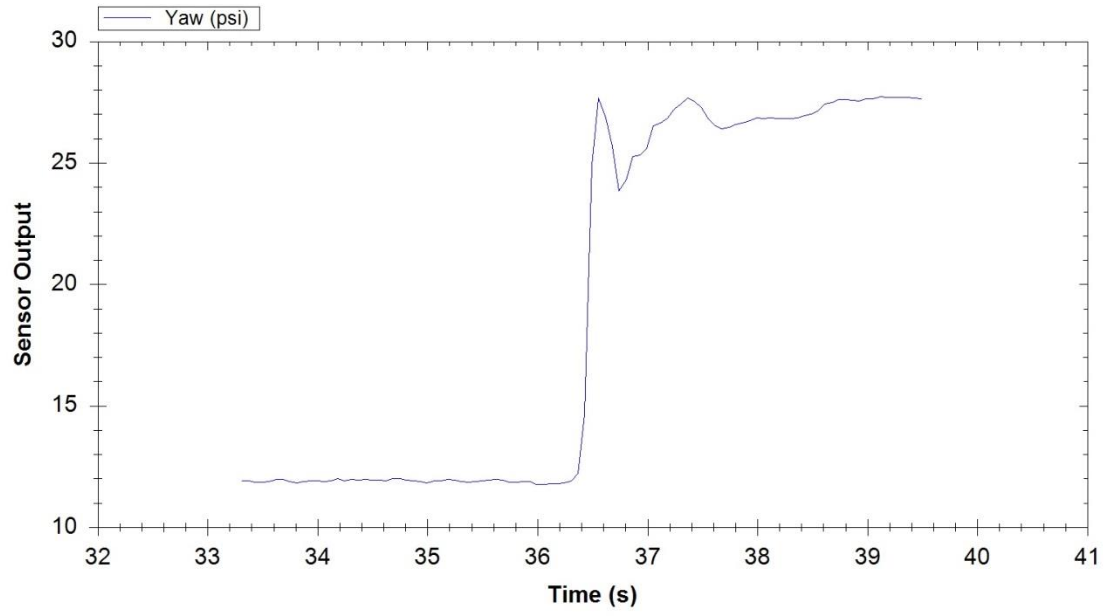


Figure 4.2- Yaw angle oscillates with lower EKF variances values

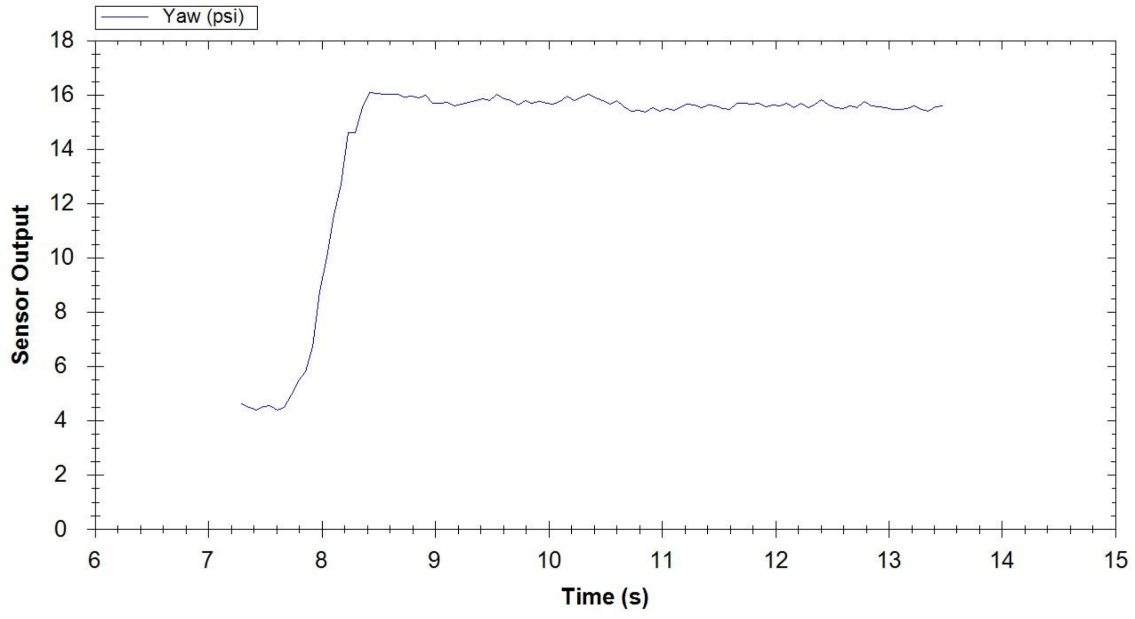


Figure 4.1-Yaw angle with calibrated EKF variances values.

4.4 Head Tracking Algorithm

After calibrating the UM6, as described in section 3.3.1.2 position data of head mounted UM6 is transferred to Arduino Mega R3, and with using this information, pan tilt roll servo motors are controlled. The sequence of rotations used to give the orientation of the UM6 is first yaw, then roll, then pitch.

At the beginning of the code, as mentioned in section 4.2.2, the request data should be transferred to UM6 since UM6 is in Listen mode. UM6 is send roll and pitch angles in the same register whereas send yaw angle alone. The address code for requesting roll and pitch angles is 0x62 where 0x63 is for yaw angle. In Listing 4 the initial values for variables and request data can be seen.

```
int RegPHITHT [7] = {0x73, 0x6E, 0x70, 0x00, 0x62, 0x01, 0xB3}; // pitch-roll
int RegPSI [7] = {0x73, 0x6E, 0x70, 0x00, 0x63, 0x01, 0xB4}; //yaw
boolean PR_OK = false;
boolean YW_OK = false;  //initial values
boolean start = true;  //for getting servos to initial position, by pressing joystick
int bufPR = 0;
int bufYW = 2;
```

Listing 4- Reaching Registers of Um6

For transferring position data from UM6, first a request must be send. After this, UM6 sends the Euler angle estimates, roll and pitch at the same time, yaw with another request. To obtain the actual angle estimate in degrees (for all roll pitch and yaw), the register data should be multiplied by the scale factor as;

$$angle\,estimate = register\_data * 0.0109863\,.$$

This conversion can be seen on Listing 5. In Table 4 and Table 5, the register data send from UM6 can be seen. At Listing 5, the Arduino code can be seen.

Table 4-Euler Angle Roll/Pitch Register Definition

B3	B2	B1	B0
Roll angle (ϕ)		Pitch angle (θ)	

Table 5-Euler Angle Yaw Register Definition

B3	B2	B1	B0
Yaw angle (ψ)		Reserved	

```
void setup()
{
   Serial2.begin(115200);
   getEulerAngle();
 }
void getEulerAngle()
{
   if ( bufPR == 0)
       sendPitchRoll();
   if ( bufYW == 0)
       sendYaw();
   if (Serial2.available() > 0)
   {
      data[data_count] = Serial2.read();
      data_count++;
   }

   if (data_count == 11)
   {
      data_count = 0;
      if (bufPR >= 5 && PR_OK == false)
          calcPitchRoll();
      if (bufYW >= 5 && YW_OK == false)
           calcYaw();
    }

   if(YW_OK == true && PR_OK == true)
   {
      if(Yaw < 0)
           Yaw = Yaw+360;
       if(start == true)
           {
              start = false;
           }
       Serial.print("ROLL UM:");
       Serial.print(Roll);
       Serial.print("   ,   ");
       Serial.print("YAW UM:");
       Serial.print(Yaw);
       Serial.print("   ,   ");
       Serial.print("Pitch UM:");
       Serial.println(Pitch);
       PR_OK = false;
       YW_OK = false;
       bufPR = 0;
       bufYW = 2;
  }
}

void sendPitchRoll()
```

```
{
   while ( bufPR <= 6 )
   {
      Serial2.write(RegPHITHT[bufPR]);
      bufPR++;
   }
}

void sendYaw()
{
    while ( bufYW <= 6 )
   {
      Serial2.write(RegPSI[bufYW]);
      bufYW++;
   }
}

void calcPitchRoll()
{
    Roll = (data[5]*255+data[6])*0.0109863;
    Pitch = (data[7]*255+data[8])*0.0109863;
    PR_OK = true;
    bufYW = 0;
}
void calcYaw()
{
    Yaw = (data[5]*255+data[6])*0.0109863;
    YW_OK = true;
}
```

Listing 5- Getiing Euler Angles from UM6

Listing 5 is the code of head tracking system. As can be seen from above, first two requests are send (roll-pitch and yaw) to UM6 for Euler Angle estimations. Then the register data is converted to actual angle estimation by multiplying scale factor at "calcPitchRol" and "calcYaw" functions. With Serial.Print command actual angle estimations are written to serial board for checking data. Finally this converted data is used for pan-tilt-roll mechanism (Listing 7).

4.5 Pan-Tilt-Roll Algorithm

After calculating real angle estimations of Euler Angles, this committed data is send to relevant servo motor for positioning. Servo.h library [40] is used for controlling servo motors. Some important parameters from this library are given in Listing 6.

```
class Servo
{
public:
  Servo();
  uint8_t attach(int pin);
  uint8_t attach(int pin, int min, int max);
  void detach();
  void write(int value);
  void writeMicroseconds(int value);
  int read();
  int readMicroseconds();
  bool attached();
private:
   uint8_t servoIndex;
   int8_t min;
   int8_t max;
}
Servo::Servo()
{
  if( ServoCount < MAX_SERVOS)
    {
     this->servoIndex = ServoCount++;
     servos[this->servoIndex].ticks = usToTicks(DEFAULT_PULSE_WIDTH);
    }
  else
    this->servoIndex = INVALID_SERVO ;
}
uint8_t Servo::attach(int pin)
{
  return this->attach(pin, MIN_PULSE_WIDTH, MAX_PULSE_WIDTH);
}
uint8_t Servo::attach(int pin, int min, int max)
{
  if(this->servoIndex < MAX_SERVOS )
    {
    pinMode( pin, OUTPUT) ;
    servos[this->servoIndex].Pin.nbr = pin;
    this->min  = (MIN_PULSE_WIDTH - min)/4;
    this->max  = (MAX_PULSE_WIDTH - max)/4;
    timer16_Sequence_t timer = SERVO_INDEX_TO_TIMER(servoIndex);
    if(isTimerActive(timer) == false)
      initISR(timer);
    servos[this->servoIndex].Pin.isActive = true;
    }
  return this->servoIndex ;
}
void Servo::detach()
{
  servos[this->servoIndex].Pin.isActive = false;
  timer16_Sequence_t timer = SERVO_INDEX_TO_TIMER(servoIndex);
  if(isTimerActive(timer) == false)
     {
      finISR(timer);
     }
}
void Servo::write(int value)
{
  if(value < MIN_PULSE_WIDTH)
  {
    if(value < 0) value = 0;
    if(value > 180) value = 180;
    value = map(value, 0, 180, SERVO_MIN(),  SERVO_MAX());
  }
  this->writeMicroseconds(value);
}
void Servo::writeMicroseconds(int value)
{
  // calculate and store the values for the given channel
  byte channel = this->servoIndex;
  if( (channel < MAX_SERVOS) )   // ensure channel is valid
  {
    if( value < SERVO_MIN() )          // ensure pulse width is valid
      value = SERVO_MIN();
    else if( value > SERVO_MAX() )
      value = SERVO_MAX();
    value = value - TRIM_DURATION;
    value = usToTicks(value);  // convert to ticks after compensating for interrupt overhead
    uint8_t oldSREG = SREG;
    cli();
    servos[channel].ticks = value;
    SREG = oldSREG;
  }
}
int Servo::read() // return the value as degrees
{
  return  map( this->readMicroseconds()+1, SERVO_MIN(), SERVO_MAX(), 0, 180);
}

int Servo::readMicroseconds()
{
  unsigned int pulsewidth;
  if( this->servoIndex != INVALID_SERVO )
    pulsewidth = ticksToUs(servos[this->servoIndex].ticks)  + TRIM_DURATION ;
  else
    pulsewidth  = 0;
  return pulsewidth;
}

bool Servo::attached()
{
  return servos[this->servoIndex].Pin.isActive ;
}
```

Listing 6-Enable to drive servo motors

In Listing 8 the code of position controlling for servo motors is given. First the initial conditions for variables are given (Listing 7).

```
int Pitch =0;
int Yaw = 0;
int Roll = 0;
int Pitch_trim = 0;
int Roll_trim = 0;
int Yaw_trim = 0;
int calPitch = 0;
int calYaw = 0;
int calRoll = 0;
byte data [15];
int data_count = 0;
Servo PitchServo;
Servo YawServo;
Servo RollServo;
```

Listing 7- Initial conditions for variables

With pressing the joystick, servo motors go to their initial position. Without this action servo motors do not start so at the start operator must press the joystick. Servo motors are connected to Arduino Mega's 2nd, 4th, and 6th pins. Pitch and roll angles start from 0^0 and end at 180^0. However, yaw angle is between 0^0-360^0. Pitch angle increases opposite of head motion so that it is needed to reverse this angle by using "map" function. Position control code for servo motors can be seen in Listing 8.

```
void setup()
{
        pinMode(JoyBtnPin,INPUT);
        digitalWrite(JoyBtnPin,HIGH);
        PitchServo.attach(2);
        YawServo.attach(6);
        RollServo.attach(4);
        PitchServo.write(90);
        YawServo.write(90);
        RollServo.write(90);
      Serial.begin(57600);
      while (digitalRead(JoyBtnPin) == true);
}
void loop()
{
        getEulerAngle();
        ReadJoystick();
}
void ReadJoystick()
{
        if(digitalRead(JoyBtnPin) == 0)
                start = true;
}
void CalibrateServo()
{
        calPitch = Pitch - 90 ;
        calYaw = Yaw - 90 ;
        calRoll = Roll - 90 ;
        ActServo();
        delay(500);
}
void ActServo()
{
        Pitch = Pitch-calPitch;
        Pitch = map ( Pitch,0,180,180,0);
        Yaw = Yaw - calYaw;
        if ( Yaw > 360 )
                Yaw = Yaw - 360;
        PitchServo.write(Pitch+Pitch_trim);
        YawServo.write(Yaw+Yaw_trim);
        RollServo.write(Roll-calRoll+Roll_trim);}
```
Listing 8-Pan-Tilt-Roll algorithm

4.6 Remote Control Algorithm

For remotely controlling the camera’s zoom in and out property, as mentioned earlier, remote codes are decoded via infrared receiver. These decoded codes are sent via infrared led to camera. The code is simple but the trick is that for Sony handycams the remote code should be sent three times. Infrared led is connected to the remote control board (Figure 3.6) and this board is fixed near to camera as the range of infrared communication is short despite inserting a transistor to remote control board for increasing the range. Zoom in and zoom out properties are controlled with the joystick (Figure 3.9), with pushing the joystick forward camera zoom out and with pushing backward, camera zoom in. Arduino code can be seen in Listing 9.

```
void IRsend::sendRaw(unsigned int buf[], int len, int hz)
{
 enableIROut(hz);
 for (int i = 0; i < len; i++) {
  if (i & 1) {
   space(buf[i]);
  }
  else {
   mark(buf[i]);
  }
 }
 space(0);
}
IRsend irsend;
unsigned int zoomIn[31] = {2400, 550, 650, 550, 1200, 550, 650, 550, 1200, 550, 1250, 550, 600,
550, 650, 550, 1200, 600, 600, 550, 650, 550, 1200, 550, 1250, 550, 600, 550, 1250, 550, 1200};
unsigned int zoomOut[31] = {2400 ,600, 1200, 500, 1250, 600, 600, 550, 1200, 550, 1250, 550,
600, 600, 600, 550, 1250, 550, 600, 550, 650, 550, 1200, 600, 1200, 500, 650, 600, 1200, 550,
1200};
void loop()
{
        ReadJoystick();
}

void ReadJoystick()
{
        Serial.print(analogRead(A0));
        int move = analogRead(A0);
        if(move < 256 ){
                irsend.sendRaw(zoomIn,31,38);
     delay(10);
     irsend.sendRaw(zoomIn,31,38);
     delay(10);
     irsend.sendRaw(zoomIn,31,38);
     delay(10);
        }
        if(move > 750 ){

                irsend.sendRaw(zoomOut,31,38);
     delay(10);
     irsend.sendRaw(zoomOut,31,38);
     delay(10);
     irsend.sendRaw(zoomOut,31,38);
     delay(10);
        }
```

Listing 9-Arduino Code of Remote Control

Joystick's analog output is cabled to Arduino Mega's A0 analog pin. If the value of A0 is smaller than 256, this means joystick is pushed backward and zoom in is made on the other when A0 is bigger than 750 joystick pushed forward, means zoom out. While the joystick is in the middle position, analog value is 512. Number smaller than 512 means joystick is pushed backward, number bigger than 512 means joystick is pushed forward. The remote codes are send via Arduino's number 9 digital pin to remote control board.

4.7 Low-Pass Filter Algorithm

For filtering noises and Euler angle drafts on Arduino side, a simple low-pass filter is added to Arduino code. This code works after receiving 40 EKF Euler data. In this code, collected data are organized in the order of biggest to smallest. 6 maximum (15% of total) and 6 minimum (15% of total) data are ignored. The rest are used for taking average value. The calculated data is send as a new position to servo motors.

```
int digitalSmooth(int rawIn, int *sensSmoothArray)
{
        int j, k, temp, top, bottom;
        long total;
        static int i;
        static int sorted[filterSamples];
        boolean done;
        i = (i + 1) % filterSamples;
        sensSmoothArray[i] = rawIn;
        for (j=0; j<filterSamples; j++)
         {
                sorted[j] = sensSmoothArray[j];
           }
        done = 0;
        while(done != 1)
         {
                 done = 1;
                 for (j = 0; j < (filterSamples - 1); j++)
                {
                           if (sorted[j] > sorted[j + 1])
                        {
                                temp = sorted[j + 1];
                                sorted [j+1] =  sorted[j] ;
                                sorted [j] = temp;
                                done = 0;
                             }
                 }
        }

        bottom = max(((filterSamples * 15)  / 100), 1);
        top = min((((filterSamples * 85) / 100) + 1  ), (filterSamples - 1));
        k = 0;
        total = 0;
        for ( j = bottom; j< top; j++)
             {
                 total += sorted[j];
                 k++;
               }
        return total / k;
}
```

Listing 10- Arduino Code of Low-Pass Filter

4.8 Conclusion

In this section, developed Arduino code for pan-tilt-roll system and for head tracking system is considered with orientation sensor UM6’s calibration. It is important to calibrate UM6 properly for collecting fine position data of fixed head. There are different ways to calibrate UM6 but the most important ones are magnetometer calibration and zeroing the rate gyros. Magnetometer calibration is done once after fixing UM6 to working platform. This calibration is done via CH Robotics Software Interface after UM6 collects minimum 300 data points. After collecting these points, the interface creates a coefficient matrix and writes this matrix to UM6’s RAM. Zeroing the rate gyros should be done each time while UM6 is powered on. This calibration can be done with external code as can be seen in Listing 3.

After fine calibration of UM6, head tracking is achieved. UM6 is used in Listen mode which means that the sensor send data after requesting. Euler angle estimations are requested and respond data is converted to actual Euler angle estimations after multiplying coming data with coefficients. Position of sensor fixed head is transferred in real time so that even movement of the head new position is known instantaneously.

With this known new positions, position control of servo motors are easily maintained. Three servo motors are controlled via Arduino Mega, and the code can be seen in Listing 8. Finally in this section the remote control of zoom in and zoom out properties of 3D camera is mentioned. For Sony cameras one should send the remote code three times for remote control function. The Arduino code is given in Listing 9.

CHAPTER 5

VALIDATION OF PAN TILT ROLL VISION SYSTEM

5.1 Introduction

After successful development and implementation of pan-tilt-roll vision system, it is needed to be validated with human participants. For this purpose two experiments were made. In first experiment Sony HMZ-T2 was used for showing the effects of 3D vision on controlling a remote control hobby car. In second experiment the differences between HMD and 6 LED monitor system was tested with 2D vision. For the experiments, a set of participants was selected, a test scenario was constituted and the tests were performed. Results were logged and evaluated after on and they are described in Results section.

For the experiments, a set of participants was selected. Remote control car setup was prepared and dependent variables were measured via surveys [41] handed out before and after experiment.

5.2 Participants

There are thirteen participants selected for the experiments. Participants are either research assistances in Mechanical Engineering Department of METU, or graduate students from the same university and Hacettepe University. Participants are males and their ages range from 24 to 29 with average value of 27.15 (SD = 1.33). 10 of 13 participants reported that they drive a car daily, 3 monthly. Five participants mentioned that they are “experts”, 6 as “excellent” with computer usage while remaining mentioned as being “good”. 9 participants reported playing video games, especially racing or driving ones.

5.3 Experiment Setup

As mentioned above, two experimental setups were set up. In the first setup differences between 2D and 3D virtual reality was tested whereas in the second the monitor systems were tested.

In each setup a remote control hobby car was used. A test area was set up with three obstacles (Figure 5.1) into the mechatronic laboratory D-107. Participants tried to control the car remotely by using vision system to perform their goals. Their goal was controlling hobby with making slalom among the obstacles which had 1.5 meters in between without crashing any of them and getting car back to start position (Figure 5.2). Pan-tilt-roll camera system was fixed back side of the setup that enables camera to capture all set up system. Participants sit a wheelchair located at left side of the camera for a HMD experiment. Camera’s HDMI output was connected to input of SONY HMZ-T2’s processor unit. Before participants started their experiment they made a calibration of HMZ-T2 for their eyes. For the 6 LED monitor experiment they sit towards the monitors. HDMI output of the camera was connected to video capture card AverTV HD.

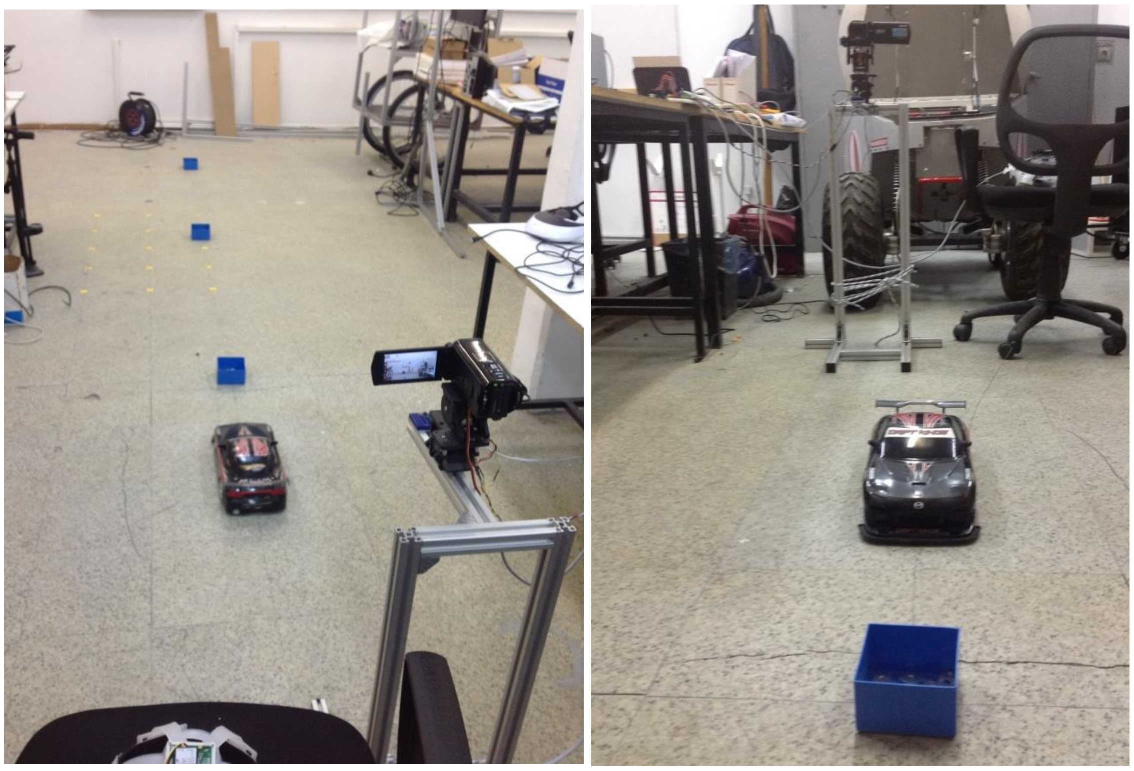


Figure 5.1-Setup test area

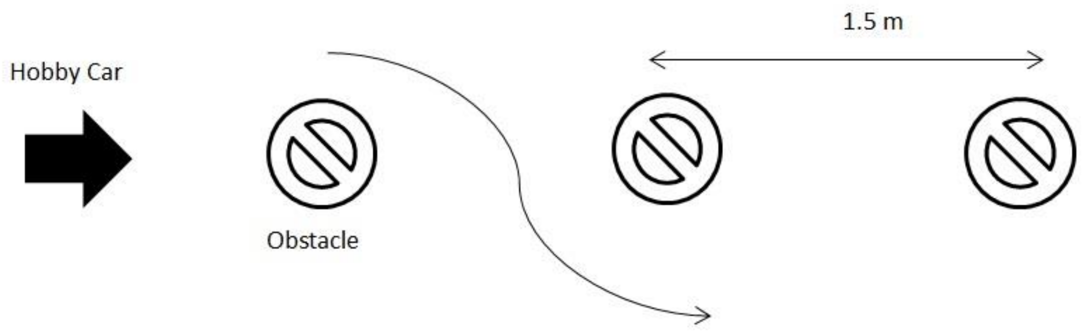


Figure 5.2-Experiment setup

5.4 Procedure

Before the experiment, participants are allowed to drive the remote control hobby car in the test route with or without line of sight to lessen learning effects. After this learning phase, demographic survey (see Appendix B) is handed out to the participant. Once the demographic survey is collected, test phase begins. First test is with HMD for the differences of 2D and 3D vision system. Participants sit a wheelchair located at left side of the camera with HMD and orientation sensor fixed hat on their head (Figure 5.3). Since they wear HMD they have no line of sight with the car. The remote control car is positioned according to the experiment setup given in Figure 5.2. It is told to the user that he is required to control the hobby car from starting point and with making slalom among to obstacles and turn back to the starting point without colliding with obstacles and wasting excessive time. After the confirmation of the participant, percipient make the calibration of HMD according to his eyes then press joystick to get servos their initial positions. After these calibrations the user is informed that he can start to mission. He guides the car using HMD in 2D at first. When the goal completed, task completion time and number of objects collided are recorded. After completion of this first phase, second phase with 3D vision is started. For this case calibration of HMD is made again. Same path is taken and again same variables are measured and recorded as in previous phase. After the end of the experiment, the participant is given a sheet of survey which is the NASA Task Load Index survey (see Appendix C) to measure the workload.

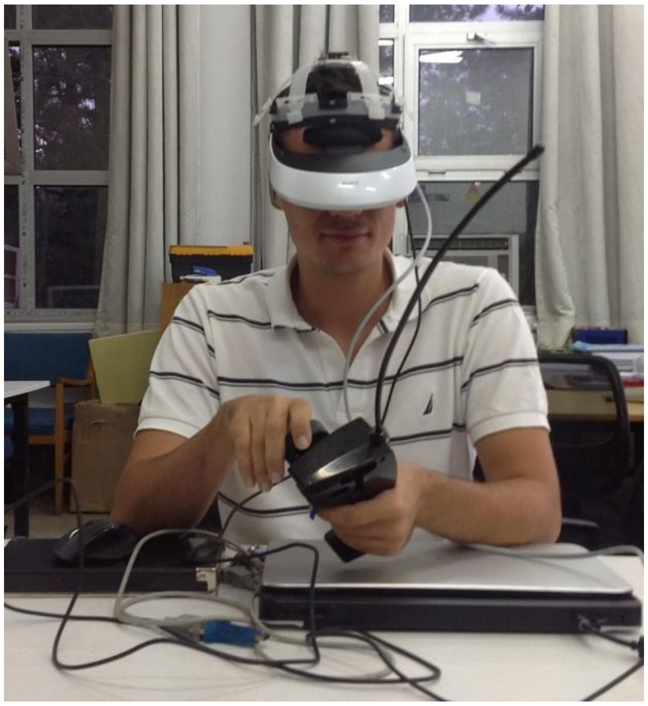


Figure 5.1-One participant in HMD test

After finishing first experiment, second experiment is made. In this experiment monitor systems, HMD and 6 LED monitors are compared. In this system the vision is 2D. Again the participants wear HMD and orientation sensor fixed helmet and complete the goal like first experiment. Same path is taken and again same variables are measured and recorded as in previous experiment. After completing this phase, participants sit towards the 6 LED monitor system with wearing orientation sensor helmet on their head. At this time they try to accomplish the goal with looking from this monitor system as seen from Figure 5.4. Again same path, same collected variables as in first phase. And also after this experiment too, the participant is given a sheet of survey which is the NASA Task Load Index survey.

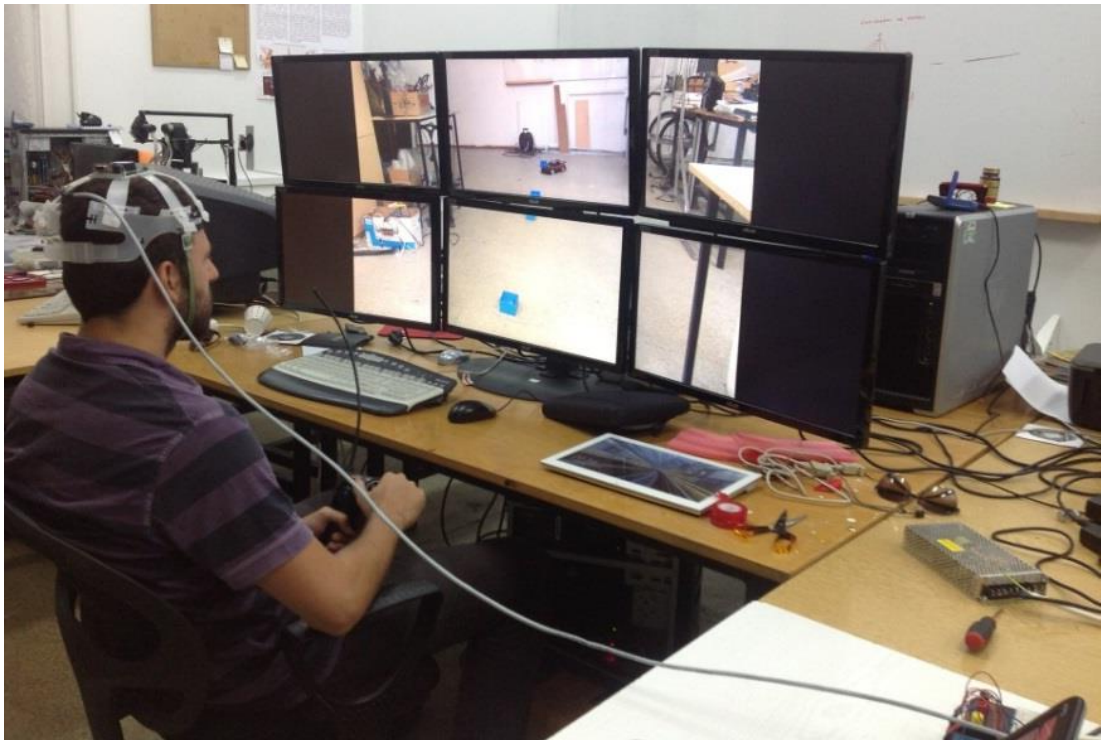


Figure 5.1- One participant in 6 LED monitor system test

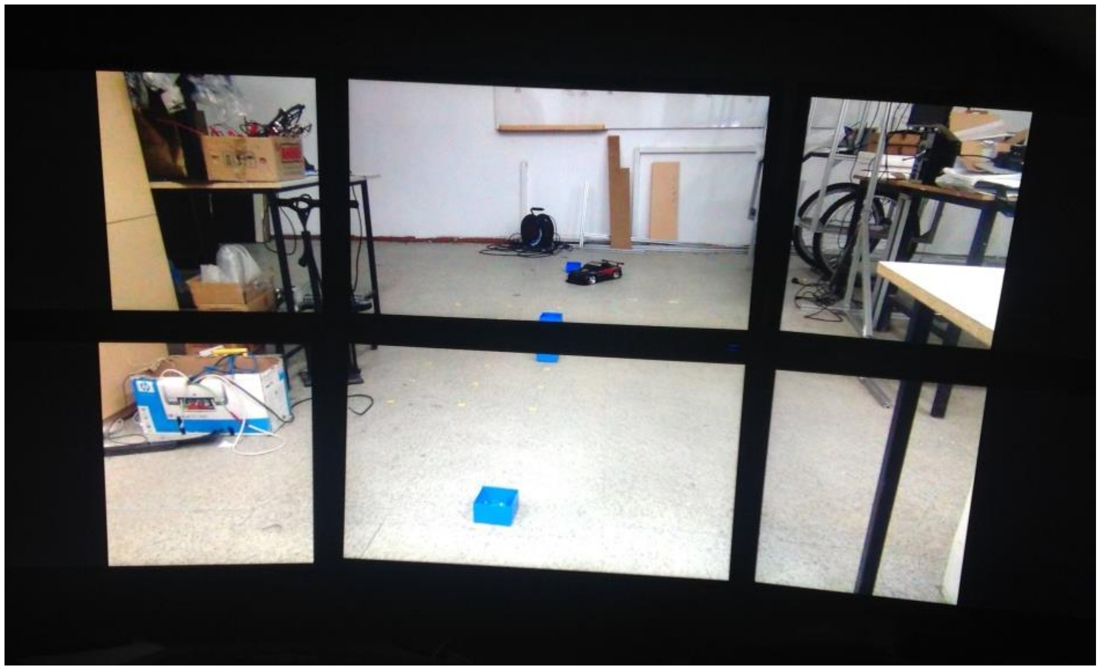


Figure 5.2-Monitor View From & LED Monitor System

NASA Task Load Index survey test procedure is applied to each participant and the resulting dependent variables are recorded after two experiments respectively. As dependent variables, task errors (number of boxes that collided), operator efficiency (time to complete task) and perceived workload (according to NASA-TLX scores) are selected.

5.5 Results

Once the tests are completed, results are collected and statistical data is generated from these results. Results section is divided into three as Task Completion Time, Number of Task Errors and Perceived Workload.

5.5.1 Task Completion Time

The primary measure is the task completion time. Time is measured from the start of the movement of the hobby car up to coming back to start point. If a time the test takes is longer than 60 seconds, the test is interrupted and the value is taken as 60. However none of the participants completed the mission beyond 60 seconds.

With 2D vision on HMD it took from 20.20 to 37.80 seconds to participants to complete the course. The mean task completion time was 28.47 seconds with a standard deviation of 4.83. Making same experiment with 3D vision on HMD time to complete the task is reduced to a mean value of 26.08 seconds and a standard deviation of 5.16. The minimum task time with for 3D vision was 16.70 seconds while the maximum was 34.60 (Table 6).

Table 6- Results of Task Completion Time For HMD experiment

Task Type	**Mean (sec)**	**St. Dev.**	**Minimum (sec)**	**Maximum (sec)**
HMD with 2D	28.47	4.83	20.20	37.80
HMD with 3D	26.08	5.16	16.70	34.60

For the second experiment, 2D vision on HMD is made again. At this time the mean is 28.68 seconds where 19.80 seconds minimum, 37.50 seconds maximum with 4.98 standard deviation. On the other side, with making same experiment 6 LED monitor system 2D vision, the mean is 27.28 seconds, the minimum is 18.50 seconds, maximum is 37.10 seconds and 5.27 is the standard deviation.

Table 7- Results of Task Completion Time For Monitor System Experiment

Task Type	**Mean (sec)**	**St. Dev.**	**Minimum (sec)**	**Maximum (sec)**
HMD with 2D	28.68	4.98	19.80	37.50
6 LED Monitor System 2D	27.28	5.27	18.50	37.10

5.5.2 Number of Task Errors

The second measure of the operator performance is the number of task errors. Task error is defined as colliding with an obstacle on the course in this experiment.

On the first experiment with 2D vision on HMD, the participants collided with total of 10 obstacles on the path. Mean value is 0.77 with a standard deviation of 0.67. On the other side, with 3D vision, 5 collisions is occurred which results in a mean of 0.38 and standard deviation of 0.47 (Table 8).

Table 8- Results of Number of Task Errors For HMD experiment

Task Type	**Mean (#)**	**St. Dev.**	**Total**
HMD with 2D	0.77	0.67	10
HMD with 3D	0.38	0.47	5

For the second experiment, with 2D vision on HMD mean is 0.62 with 8 collisions. The standard deviation is 0.60. On the other side, with 2D vision on 6 LED monitor system, 7 collisions occurs which means 0.54 mean with 0.61 standard deviation.

Table 9- Results of Number of Task Errors For Monitor System Experiment

Task Type	**Mean (#)**	**St. Dev.**	**Total**
HMD with 2D	0.62	0.60	8
6 LED Monitor System 2D	0.54	0.61	7

5.5.3 Perceived Workload

Perceived workload of the operators is the final dependent variable to be measured. To measure the workload properly, Task Load Index of NASA (Appendix C) is used in both types of task.

At first experiment it is seen that mental demand is more in the case of 2D. However physical demand is nearly same probably because head tracking system is same in both phases. Participants feel more frustrated when 2D vision on screens. Time pressure is less with 2D vision and the participants feel that they are more successful with the 3D vision support in accordance with the previous results in task completion time and task errors (Table 10).

Table 10- Results of NASA Task Load Index For HMD experiment

Task Type	Mental Demand	Physical Demand	Temporal Demand	Level of Effort	Level of Frustration	Perform.
HMD with 2D	6.46	4.00	3.46	5.77	7.31	7.23
HMD with 3D	5.77	3.85	5.08	4.23	6.69	8.92

On the second experiment, for testing monitor systems, mental demand and physical demand is nearly same for 6 LED monitor system and HMD with 2D vision. 6 LED monitor system has a better performance index and participants feel less frustrated with monitor system (Table 11).

Table 11- Results of NASA Task Load Index For Monitor System Experiment

Task Type	Mental Demand	Physical Demand	Temporal Demand	Level of Effort	Level of Frustration	Perform.
HMD with 2D	6.23	3.92	3.54	5.92	7.54	7.00
6 LED Monitor System 2D	4.92	3.54	3.38	5.54	7.23	7.69

5.6 Conclusion

In this chapter, with the experiments, effects of 3D vision and big monitor system are tested. Sony HMZ-T2 HMD and 6 LED monitor system is used in this experiments.

Experiment results mentioned in previous sections show that the task completion time is shortened and task errors are lessened with introduction of the 3D vision. In addition to this for 2D vision, participants are more easily adapted 6 LED monitor system than HMD. Results of perceived task load indicate that, 3D vision and big screens can be beneficial in terms of operator workload. Also these results validate that with the developed system participants do not have any motion sickness.

CHAPTER 6

CONCLUSION & FUTURE WORK

6.1 Conclusion

In this thesis pan tilt roll vision system is developed and validation of vision systems is tested with two different experiments. System consists of 3 digital coreless servo motors for pan tilt roll mechanism, Sony HDR-TD30 3D handycam, UM6 orientation sensor fixed wearable plastic helmet, and for monitoring Sony HMZ-T2 HMD and 6 LED monitor display system.

After introduction part a literature survey is conducted on the similar projects in the literature. This part is divided into two parts; pan-tilt roll system and head tracking parts. In each parts, mechanical concerns, hardware, software the aim of project's, where to use are considered.

In the third part, the equipment that is used in this project is considered. Technical details, specifications, choosing reasons are considered. In the next section the developed algorithm is mentioned.

For validating the system works, two different experiments are done with 13 participants. 2 different monitor systems are prepared in this thesis. In one experiment the differences between 2D and 3D vision is tested with controlling remote control hobby car. First the participants drive the car without looking any monitor for minimizing the learning effect. It is seen that for participants it is easier to adapted 3D vision on HMD. In other experiment HMD and 6 monitors display system is tested. In this test participants are more adaptive to 6 monitors system. But from the results it is seen that 3D HMD is the best for vision system.

6.2 Future Work

In this study as mentioned before Digital coreless RC servo motors are used. As the most important future work these motors are replaced with high quality zero backlash encoder motors, such as Faulhaber or Maxon. With changing these motors more stable system can be obtained.

This system is thought to use in our Tübitak supported project, Keçi. So in future, the system is adapted to Keçi for improving operator's performance.

Also due to lack of 3D active glasses, 3D performance comparison of HMD and monitor system is not achieved. After bringing glasses, this test will be completed.

REFERENCES

[1] Michael A. Goodrich and Alan C. Schultz, "Human–Robot Interaction: A Survey," *Human–Computer Interaction*, vol. 1, no. 3, pp. 203-275, 2007.

[2] M.C Cavusoglu, A. Sherman, and F. Tendick, "Design of bilateral teleoperation controllers for haptic exploration and telemanipulation of soft environments," *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 4, pp. 641-647, August 2002.

[3] Pattie Maes Pranav Mistry, "SixthSense: a wearable gestural interface," in *ACM SIGGRAPH ASIA* , New York, NY, USA, 2009.

[4] Sony.com, Last accessed in August, 2013. Sony PTZ Camera Image. [Online]. http://pro.sony.com/bbsc/ssr/cat-broadcastcameras/cat-broadcastcamerapantiltzoom/product-BRCH900/

[5] Panasonic.com, Last accessed in August, 2013. Panasonic PTZ Camera Image. [Online]. http://www.panasonic.com/business/provideo/AW-HE120.asp#

[6] Sebastian Lang et al., "Providing the Basis for Human-Robot-Interaction:A Multi-Modal Attention System for a Mobile Robot," in *International Conference on Multimodal Interfaces (ICMI)*, Vancouver, 2003, pp. 28-35.

[7] BIRON, Last accessed in August, 2013. BIRON Image. [Online]. http://aiweb.techfak.uni-bielefeld.de/files/old-site/projects/BIRON/welcome.html

[8] Greg Kogut, Dr. Mike Blackburn, and H.R. Everett. (2003, December) Using Video Sensor Networks to Command and Control Unmanned Ground Vehicles. [Online]. http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA422062

[9] S.N. Fry, M. Bichsel, P. Müller, and D. Robert, "Tracking of flying insects using pan-tilt cameras," *Journal of Neuroscience Methods*, vol. 101, pp. 59-67, 2000.

[10] Sudipta N Sinha and Marc Pollefeys, "Towards Calibrating a Pan-Tilt-Zoom Camera Network," in *Workshop on Omnidirectional Vision and Camera Networks held in conjunction with ECCV 2004*, OMNIVIS, 2004.

[11] Tie Hu, Peter K. Allen, and Nancy J. Hogle and Dennis L. Fowler, "Insertable Surgical Imaging Device with Pan, Tilt, Zoom, and Lighting," *The International Journal of Robotics Research*, vol. 28, no. 10, pp. 1373-1386, October 2009.

[12] Samer Abdallah , Sebastien Rougeaux , Alexander Zelinsky Harley Truong, "A Novel Mechanism for Stereo Active Vision," in *IN PROC. AUSTRALIAN CONFERENCE ON ROBOTICS AND AUTOMATION*, 2000.

[13] Ole C. Jakobsen and Eric N. Johnson, "Control Architecture for a UAV-Mounted Pan/Tilt/Roll Camera Gimbal," in *American Institute of Aeronautics and Astronautics*, Arlington, Virginia, 2005.

[14] DongBin Lee, Timothy C. Burg, and Darren M. Dawson and Guenther Dorn, "Fly-the-Camera Perspective: Control of a Remotely Operated Quadrotor UAV and Camera Unit," in *Aerial Vehicles*, Thanh Mung, Ed.: InTech, 2009, ch. 8, pp. 161-188.

[15] Foxlin E. and Harrington M., "WearTrack: a self-referenced head and hand tracker for wearable computers and portable VR," in *Wearable Computers, The Fourth International Symposium on*, Atlanta, GA, USA, October 2000, pp. 155 - 162.

[16] Jacque M. Joffrion, *HEAD TRACKING FOR 3D AUDIO USING A GPS-AIDED MEMS IMU*. DEPARTMENT OF THE AIR FORCE AIR UNIVERSITY, Ohio, M.S. Thesis, 2009.

[17] Eric Foxlin and Leonid Naimark, "VIS-Tracker: A Wearable Vision-Inertial Self-Tracker," in *IEEE Virtual Reality Conference 2003 (VR 2003)*, Los Angeles, CA, USA, 22-26 March, 2003.

[18] Institute of Electrical Measurement, Graz, Last accessed in August 2013. [Online]. http://www.emt.tugraz.at/~tracking/topics.htm

[19] Eric. Foxlin, "Inertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter.," in *Proc. of IEEE Virtual Reality Annual International Symposium.*, 1996, pp. 184-194.

[20] Michael Harrington and Yury Altshuler Eric Foxlin, "Miniature 6-DOF inertial system for tracking HMDs," in *Helmet and Head-Mounted Displays III*, Orlando, April 13-14 1998.

[21] Eric Foxlin, "Head-tracking relative to a moving vehicle or simulator platform," in *Proceedings of Helmet and Head-Mounted Displays V*, Orlando, April 24-25, 2000.

[22] Xinyu Xu and Baoxin Li, "Head Tracking Using Particle Filter with Intensity Gradient and Color Histogram," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference*, Amsterdam, 6-6 July 2005, pp. 888 - 891.

[23] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," in *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference*, Santa Barbara, CA, 23-25 Jun 1998, pp. 232 - 237.

[24] Ce Wang and Michael S. Brandstein, "A HYBRID REAL-TIME FACE TRACKING SYSTEM," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference*, Seattle, WA, 12-15 May 1998, pp. 3737 - 3740 vol.6.

[25] Craig Hennessey, Borna Noureddin, and Peter Lawrence, "A single camera eye-gaze tracking system with free head motion," in *Eye tracking research & applications 2006, ETRA '06*, New York, NY, USA, 2006, pp. 87-94.

[26] Servo City: Pan-Tilt mechanism. Image. Last accessed in August, 2013. [Online]. http://www.servocity.com/html/spt200_pan___tilt_system.html

[27] Servodatabase: Hitec HS-5955TG - Ultra Torque Servo. Image. Last accessed in August, 2013. [Online]. http://www.servodatabase.com/servo/hitec/hs-5955tg

[28] Savoxusa: Savox SC-1256TG High Torque Titanium Gear Digital Servo. Image. Last accessed in August,. [Online]. http://www.savoxusa.com/Savox_SC1256TG_Digital_Servo_p/savsc1256tg.htm

[29] SONY: Full HD 3D Camcorder HDRTD30V. Image. Last accessed in August, 2013. [Online]. http://store.sony.com/p/Sony-Handycam-Full-HD-3D-Camcorder/en/p/HDRTD30V

[30] SONY: HMZ-T2. Image. Last accessed in August, 2013.. [Online]. http://www.sony.com.tr/product/kisisel-3d-goruntuleyicisi/hmz-t2

[31] ASUS: ASUSHD7970. Image. Last accessed in August, 2013.. [Online]. http://www.asus.com/Graphics_Cards/HD7970DC2T3GD5/

[32] Video Capture Card: Avertv HD Video Capture Card. Last accessed on August 2013. Image.. [Online]. http://www.amazon.com/Avertv-Definition-Analog-Capture-MTVHDDVRR/dp/B002SQE1O0

[33] UM6: CHR-UM6 Orientation Sensor. Image. Last accessed in August, 2013. [Online].

http://www.roboweb.net/rw-pl-1255.html

[34] Phil W. McClureb, Lori A. Michenerc Andrew R. Kardunaa, "Scapular kinematics: effects of altering the Euler angle sequence of rotations," *Journal of Biomechanics*, vol. 33, no. 9, pp. 1063–1068, September 2000.

[35] JJE Slotine H Asada, "Kinematics in Geometry," in *Robot analysis and control*. New York, USA: A Wiley-Interscience Publication, 1986, ch. 2, pp. 23-27.

[36] V. Zatsiorsky, *Kinematics of Human Motion*, 1st ed., Lynn M. Hooper, Ed. Champaign, Illinois, USA: Human Kinetics Publishers, 1998.

[37] Whitestone JJ Albery CB, "Medical Imaging. Proc.," *Comparison of Cadaveric Human Head Mass Properties:Mechanical Measurement and Calculation*, pp. 157-171, 2003.

[38] Simon Haykin, "Kalman Filters," in *KALMAN FILTERING AND NEURAL NETWORKS*, Simon Haykin, Ed. New York, USA: John Wiley & Sons, Inc., 2001, ch. 1, pp. 1-20.

[39] Controller Board: Arduino Mega R3. Image. Last accessed in August, 2013.. [Online]. http://www.roboweb.net/rw-el-67.html

[40] Servo motor library for Arduino. Last Accessed on August 2013.. zip file. [Online]. http://playground.arduino.cc/ComponentLib/servo

[41] Akif Hacınecipoğlu, *DEVELOPMENT OF ELECTRICAL AND CONTROL SYSTEM OF AN UNMANNED GROUND VEHICLE FOR FORCE FEEDBACK TELEOPERATION*. Ankara: METU Mechanical Engiinering Master Theisis, 2012.

[42] Monitor Stand: Six LED monitor stand. Image. Last accessed in August, 2013.. [Online]. http://www.serend.com.tr/ccediloklu-monitor-standlar305.html

[43] Inertial Frame: Roll-Pitch-Yaw Angles. Image. Last accessed in August, 2013. [Online]. [Online]. http://cog.yonsei.ac.kr/quad/quad.htm

APPENDIX A

DEMOGRAPHIC SURVEY

Participant #: Age: Gender: Male / Female Date:

1. How often do you

Drive a car?

Daily	Weekly	Monthly
Once every few months	Rarely	Never

Use a joystick/steering wheel?

Daily	Weekly	Monthly
Once every few months	Rarely	Never

Play computer/video games?

Daily	Weekly	Monthly
Once every few months	Rarely	Never

2. Which type(s) of computer/video games do you most often play if you play at least once every few months?

3. Which of the following best describes your expertise with computer? (Check one)

_____ Novice

_____ Good with one type of software package (such as word processing or slides)

_____ Good with several software packages

_____ Can program in one language and use several software packages

_____ Can program in several languages and use several software packages

4. Are you in your usual state of health physically? YES NO

If NO, please briefly explain:

5. How many hours of sleep did you get last night? ______ hours

6. Do you have normal color vision? YES NO

7. Do you have prior military service? YES NO If Yes, how long ___________

APPENDIX B

NASA-TLX QUESTIONNAIRE

Please rate your **overall** impression of demands imposed on you during the exercise.

1. Mental Demand: How much mental and perceptual activity was required (e.g., thinking, looking, searching, etc.)? Was the task easy or demanding, simple or complex, exacting or forgiving?

LOW |---|---|---|---|---|---|---|---|---|---| HIGH
1 2 3 4 5 6 7 8 9 10

2. Physical Demand: How much physical activity was required (e.g., pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?

LOW |---|---|---|---|---|---|---|---|---|---| HIGH
1 2 3 4 5 6 7 8 9 10

3. Temporal Demand: How much time pressure did you feel due to the rate or pace at which the task or task elements occurred? Was the pace slow and leisurely or rapid and frantic?

LOW |---|---|---|---|---|---|---|---|---|---| HIGH
1 2 3 4 5 6 7 8 9 10

4. Level of Effort: How hard did you have to work (mentally and physically) to accomplish your level of performance?

LOW |---|---|---|---|---|---|---|---|---|---| HIGH
1 2 3 4 5 6 7 8 9 10

5. Level of Frustration: How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

LOW |---|---|---|---|---|---|---|---|---|---| HIGH
1 2 3 4 5 6 7 8 9 10

6. Performance: How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?

LOW |---|---|---|---|---|---|---|---|---|---| HIGH
1 2 3 4 5 6 7 8 9 10

APPENDIX C

LIBRARY REFERENCE

```
#ifndef Servo_h
#define Servo_h
#include <inttypes.h>
/*
 * Defines for 16 bit timers used with  Servo library
 *
 * If _useTimerX is defined then TimerX is a 16 bit timer on the curent board
 * timer16_Sequence_t enumerates the sequence that the timers should be
allocated
 * _Nbr_16timers indicates how many 16 bit timers are available.
 *
 */
// Say which 16 bit timers can be used and in what order
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
#define _useTimer5
#define _useTimer1
#define _useTimer3
#define _useTimer4
typedef enum { _timer5, _timer1, _timer3, _timer4, _Nbr_16timers }
timer16_Sequence_t ;
#elif defined(__AVR_ATmega32U4__)
#define _useTimer3
#define _useTimer1
typedef enum { _timer3, _timer1, _Nbr_16timers } timer16_Sequence_t ;
#elif defined(__AVR_AT90USB646__) || defined(__AVR_AT90USB1286__)
#define _useTimer3
#define _useTimer1
typedef enum { _timer3, _timer1, _Nbr_16timers } timer16_Sequence_t ;
#elif defined(__AVR_ATmega128__)
||defined(__AVR_ATmega1281__)||defined(__AVR_ATmega2561__)
#define _useTimer3
#define _useTimer1
typedef enum { _timer3, _timer1, _Nbr_16timers } timer16_Sequence_t ;
#else  // everything else
#define _useTimer1
typedef enum { _timer1, _Nbr_16timers } timer16_Sequence_t ;
#endif
#define Servo_VERSION           2      // software version of this library
#define MIN_PULSE_WIDTH       544     // the shortest pulse sent to a servo
#define MAX_PULSE_WIDTH      2400     // the longest pulse sent to a servo
#define DEFAULT_PULSE_WIDTH  1500     // default pulse width when servo is
attached
#define REFRESH_INTERVAL    20000     // minumim time to refresh servos in
microseconds
#define SERVOS_PER_TIMER       12     // the maximum number of servos
controlled by one timer
#define MAX_SERVOS   (_Nbr_16timers  * SERVOS_PER_TIMER)
#define INVALID_SERVO         255     // flag indicating an invalid servo
```

```
index
typedef struct  {
  uint8_t nbr        :6 ;             // a pin number from 0 to 63
  uint8_t isActive   :1 ;             // true if this channel is enabled, pin
not pulsed if false
} ServoPin_t   ;
typedef struct {
  ServoPin_t Pin;
  unsigned int ticks;
} servo_t;
class Servo
{
public:
  Servo();
  uint8_t attach(int pin);           // attach the given pin to the next free
channel, sets pinMode, returns channel number or 0 if failure
  uint8_t attach(int pin, int min, int max); // as above but also sets min and
max values for writes.
  void detach();
  void write(int value);             // if value is < 200 its treated as an
angle, otherwise as pulse width in microseconds
  void writeMicroseconds(int value); // Write pulse width in microseconds
  int read();                        // returns current pulse width as an
angle between 0 and 180 degrees
  int readMicroseconds();            // returns current pulse width in
microseconds for this servo (was read_us() in first release)
  bool attached();                   // return true if this servo is attached,
otherwise false
private:
   uint8_t servoIndex;               // index into the channel data for this
servo
   int8_t min;                       // minimum is this value times 4 added to
MIN_PULSE_WIDTH
   int8_t max;                       // maximum is this value times 4 added to
MAX_PULSE_WIDTH
};
#endif
```

/****************** ******************************/

```
#include <avr/interrupt.h>
#include <WProgram.h>
#include "Servo.h"
#define usToTicks(_us)    (( clockCyclesPerMicrosecond()* _us) / 8)     //
converts microseconds to tick (assumes prescale of 8)  // 12 Aug 2009
#define ticksToUs(_ticks) (( (unsigned)_ticks * 8)/
clockCyclesPerMicrosecond() ) // converts from ticks back to microseconds
#define TRIM_DURATION       2                               // compensation
ticks to trim adjust for digitalWrite delays // 12 August 2009
//#define NBR_TIMERS        (MAX_SERVOS / SERVOS_PER_TIMER)
static servo_t servos[MAX_SERVOS];                          // static array of
servo structures
static volatile int8_t Channel[_Nbr_16timers ];             // counter for the
servo being pulsed for each timer (or -1 if refresh interval)
uint8_t ServoCount = 0;                                     // the total
number of attached servos
// convenience macros
```

```
#define SERVO_INDEX_TO_TIMER(_servo_nbr) ((timer16_Sequence_t)(_servo_nbr /
SERVOS_PER_TIMER)) // returns the timer controlling this servo
#define SERVO_INDEX_TO_CHANNEL(_servo_nbr) (_servo_nbr %
SERVOS_PER_TIMER)       // returns the index of the servo on this timer
#define SERVO_INDEX(_timer,_channel)  ((_timer*SERVOS_PER_TIMER) +
_channel)     // macro to access servo index by timer and channel
#define
SERVO(_timer,_channel)  (servos[SERVO_INDEX(_timer,_channel)])            //
macro to access servo class by timer and channel
#define SERVO_MIN() (MIN_PULSE_WIDTH - this->min * 4)  // minimum value in uS
for this servo
#define SERVO_MAX() (MAX_PULSE_WIDTH - this->max * 4)  // maximum value in uS
for this servo
/************ static functions common to all instances
***********************/
static inline void handle_interrupts(timer16_Sequence_t timer, volatile
uint16_t *TCNTn, volatile uint16_t* OCRnA)
{
  if( Channel[timer] < 0 )
    *TCNTn = 0; // channel set to -1 indicated that refresh interval completed
so reset the timer
  else{
    if( SERVO_INDEX(timer,Channel[timer]) < ServoCount &&
SERVO(timer,Channel[timer]).Pin.isActive == true )
      digitalWrite( SERVO(timer,Channel[timer]).Pin.nbr,LOW); // pulse this
channel low if activated
  }
  Channel[timer]++;    // increment to the next channel
  if( SERVO_INDEX(timer,Channel[timer]) < ServoCount && Channel[timer] <
SERVOS_PER_TIMER) {
    *OCRnA = *TCNTn + SERVO(timer,Channel[timer]).ticks;
    if(SERVO(timer,Channel[timer]).Pin.isActive == true)     // check if
activated
      digitalWrite( SERVO(timer,Channel[timer]).Pin.nbr,HIGH); // its an
active channel so pulse it high
  }
  else {
    // finished all channels so wait for the refresh period to expire before
starting over
    if( (unsigned)*TCNTn <  (usToTicks(REFRESH_INTERVAL) + 4) )  // allow a
few ticks to ensure the next OCR1A not missed
      *OCRnA = (unsigned int)usToTicks(REFRESH_INTERVAL);
    else
      *OCRnA = *TCNTn + 4;  // at least REFRESH_INTERVAL has elapsed
    Channel[timer] = -1; // this will get incremented at the end of the
refresh period to start again at the first channel
  }
}
#ifndef WIRING // Wiring pre-defines signal handlers so don't define any if
compiling for the Wiring platform
// Interrupt handlers for Arduino
#if defined(_useTimer1)
SIGNAL (TIMER1_COMPA_vect)
{
  handle_interrupts(_timer1, &TCNT1, &OCR1A);
}
#endif
#if defined(_useTimer3)
SIGNAL (TIMER3_COMPA_vect)
```

```
{
  handle_interrupts(_timer3, &TCNT3, &OCR3A);
}
#endif
#if defined(_useTimer4)
SIGNAL (TIMER4_COMPA_vect)
{
  handle_interrupts(_timer4, &TCNT4, &OCR4A);
}
#endif
#if defined(_useTimer5)
SIGNAL (TIMER5_COMPA_vect)
{
  handle_interrupts(_timer5, &TCNT5, &OCR5A);
}
#endif
#elif defined WIRING
// Interrupt handlers for Wiring
#if defined(_useTimer1)
void Timer1Service()
{
  handle_interrupts(_timer1, &TCNT1, &OCR1A);
}
#endif
#if defined(_useTimer3)
void Timer3Service()
{
  handle_interrupts(_timer3, &TCNT3, &OCR3A);
}
#endif
#endif
static void initISR(timer16_Sequence_t timer)
{
#if defined (_useTimer1)
  if(timer == _timer1) {
    TCCR1A = 0;             // normal counting mode
    TCCR1B = _BV(CS11);     // set prescaler of 8
    TCNT1 = 0;              // clear the timer count
#if defined(__AVR_ATmega8__)|| defined(__AVR_ATmega128__)
    TIFR |= _BV(OCF1A);      // clear any pending interrupts;
    TIMSK |=  _BV(OCIE1A) ;  // enable the output compare interrupt
#else
    // here if not ATmega8 or ATmega128
    TIFR1 |= _BV(OCF1A);     // clear any pending interrupts;
    TIMSK1 |=  _BV(OCIE1A) ; // enable the output compare interrupt
#endif
#if defined(WIRING)
    timerAttach(TIMER1OUTCOMPAREA_INT, Timer1Service);
#endif
  }
#endif
#if defined (_useTimer3)
  if(timer == _timer3) {
    TCCR3A = 0;             // normal counting mode
    TCCR3B = _BV(CS31);     // set prescaler of 8
    TCNT3 = 0;              // clear the timer count
#if defined(__AVR_ATmega128__)
    TIFR |= _BV(OCF3A);     // clear any pending interrupts;
        ETIMSK |= _BV(OCIE3A);  // enable the output compare interrupt
#else
    TIFR3 = _BV(OCF3A);     // clear any pending interrupts;
    TIMSK3 =  _BV(OCIE3A) ; // enable the output compare interrupt
#endif
#if defined(WIRING)
    timerAttach(TIMER3OUTCOMPAREA_INT, Timer3Service);  // for Wiring platform
only
#endif
  }
#endif
#if defined (_useTimer4)
  if(timer == _timer4) {
    TCCR4A = 0;             // normal counting mode
    TCCR4B = _BV(CS41);     // set prescaler of 8
    TCNT4 = 0;              // clear the timer count
    TIFR4 = _BV(OCF4A);     // clear any pending interrupts;
    TIMSK4 =  _BV(OCIE4A) ; // enable the output compare interrupt
  }
#endif
#if defined (_useTimer5)
  if(timer == _timer5) {
    TCCR5A = 0;             // normal counting mode
    TCCR5B = _BV(CS51);     // set prescaler of 8
    TCNT5 = 0;              // clear the timer count
    TIFR5 = _BV(OCF5A);     // clear any pending interrupts;
    TIMSK5 =  _BV(OCIE5A) ; // enable the output compare interrupt
  }
#endif
}
static void finISR(timer16_Sequence_t timer)
{
    //disable use of the given timer
#if defined WIRING   // Wiring
  if(timer == _timer1) {
    #if defined(__AVR_ATmega1281__)||defined(__AVR_ATmega2561__)
    TIMSK1 &=  ~_BV(OCIE1A) ;  // disable timer 1 output compare interrupt
    #else
    TIMSK &=  ~_BV(OCIE1A) ;  // disable timer 1 output compare interrupt
    #endif
    timerDetach(TIMER1OUTCOMPAREA_INT);
  }
  else if(timer == _timer3) {
    #if defined(__AVR_ATmega1281__)||defined(__AVR_ATmega2561__)
    TIMSK3 &= ~_BV(OCIE3A);    // disable the timer3 output compare A
interrupt
    #else
    ETIMSK &= ~_BV(OCIE3A);    // disable the timer3 output compare A
interrupt
    #endif
    timerDetach(TIMER3OUTCOMPAREA_INT);
  }
#else
    //For arduino - in future: call here to a currently undefined function to
reset the timer
#endif
}
static boolean isTimerActive(timer16_Sequence_t timer)
{
  // returns true if any servo is active on this timer
  for(uint8_t channel=0; channel < SERVOS_PER_TIMER; channel++) {
    if(SERVO(timer,channel).Pin.isActive == true)
      return true;
  }
  return false;
}
/****************** end of static functions ******************************/
Servo::Servo()
{
  if( ServoCount < MAX_SERVOS) {
    this->servoIndex = ServoCount++;                    // assign a servo
index to this instance
        servos[this->servoIndex].ticks = usToTicks(DEFAULT_PULSE_WIDTH);   //
store default values  - 12 Aug 2009
  }
  else
    this->servoIndex = INVALID_SERVO ;  // too many servos
}
uint8_t Servo::attach(int pin)
{
  return this->attach(pin, MIN_PULSE_WIDTH, MAX_PULSE_WIDTH);
}
uint8_t Servo::attach(int pin, int min, int max)
{
  if(this->servoIndex < MAX_SERVOS ) {
    pinMode( pin, OUTPUT) ;                                   // set servo pin
to output
    servos[this->servoIndex].Pin.nbr = pin;
    // todo min/max check: abs(min - MIN_PULSE_WIDTH) /4 < 128
    this->min  = (MIN_PULSE_WIDTH - min)/4; //resolution of min/max is 4 uS
    this->max  = (MAX_PULSE_WIDTH - max)/4;
    // initialize the timer if it has not already been initialized
    timer16_Sequence_t timer = SERVO_INDEX_TO_TIMER(servoIndex);
    if(isTimerActive(timer) == false)
      initISR(timer);
    servos[this->servoIndex].Pin.isActive = true;  // this must be set after
the check for isTimerActive
  }
  return this->servoIndex ;
}
void Servo::detach()
{
  servos[this->servoIndex].Pin.isActive = false;
  timer16_Sequence_t timer = SERVO_INDEX_TO_TIMER(servoIndex);
  if(isTimerActive(timer) == false) {
    finISR(timer);
  }
}
void Servo::write(int value)
{
  if(value < MIN_PULSE_WIDTH)
  {  // treat values less than 544 as angles in degrees (valid values in
microseconds are handled as microseconds)
    if(value < 0) value = 0;
    if(value > 180) value = 180;
    value = map(value, 0, 180, SERVO_MIN(),  SERVO_MAX());
  }
  this->writeMicroseconds(value);
}
```

```
void Servo::writeMicroseconds(int value)
{
  // calculate and store the values for the given channel
  byte channel = this->servoIndex;
  if( (channel >= 0) && (channel < MAX_SERVOS) )   // ensure channel is valid
  {
    if( value < SERVO_MIN() )          // ensure pulse width is valid
      value = SERVO_MIN();
    else if( value > SERVO_MAX() )
      value = SERVO_MAX();

        value = value - TRIM_DURATION;
    value = usToTicks(value);  // convert to ticks after compensating for
interrupt overhead - 12 Aug 2009
    uint8_t oldSREG = SREG;
    cli();
    servos[channel].ticks = value;
    SREG = oldSREG;
  }
}
int Servo::read() // return the value as degrees
{
  return  map( this->readMicroseconds()+1, SERVO_MIN(), SERVO_MAX(), 0,
180);
}
int Servo::readMicroseconds()
{
  unsigned int pulsewidth;
  if( this->servoIndex != INVALID_SERVO )
    pulsewidth = ticksToUs(servos[this->servoIndex].ticks)  + TRIM_DURATION
;   // 12 aug 2009
  else
    pulsewidth  = 0;
  return pulsewidth;
}
bool Servo::attached()
{
  return servos[this->servoIndex].Pin.isActive ;
}
```