Development of a Night Vision Goggle Heads Up Display For Paratrooper Guidance

Fernando Ontiveros

Development of a Night Vision Goggle Heads-Up Display For Paratrooper Guidance

THESIS

Fernando Ontiveros, Captain, USAF

AFIT/GCS/ENG/08-24

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/08-24

Development of a Night Vision Goggle Heads-Up Display For Paratrooper Guidance

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science (Computer Science)

Fernando Ontiveros, B.S.C.S., M.S.C.S.

Captain, USAF

June 2008

AFIT/GCS/ENG/08-24

Development of a Night Vision Goggle Heads-Up Display For Paratrooper Guidance

Fernando Ontiveros, B.S.C.S., M.S.C.S.

Captain, USAF

Approved:

/signed/	28 May 2008
Dr. J. Raquet, (Chairman)	date
/signed/	28 May 2008
Lt Col J. McDonald, PhD (Member)	date
/signed/	28 May 2008
Lt Col S. Kurkowski, PhD (Member)	date

AFIT/GCS/ENG/08-24

Abstract

This thesis provides the proof of concept for the development and implementation of a Global Positioning System (GPS) display via Night Vision Goggles (NVG) Heads-Up Display (HUD) for paratroopers. The system has been designed for soldiers who will be able to utilize the technology in the form of a processing system worn in an ammo pouch and displayed via NVG HUD as a tunnel in the sky. The tunnel in the sky display design is essentially a series of boxes displayed within the goggle's HUD leading the paratrooper to the desired Landing Zone (LZ). The algorithm developed receives GPS and inertial sensor data (both position and attitude), and displays the guidance information in the paratrooper's NVG HUD as the tunnel in the sky. The primary goal of the project is to provide a product which allows military personnel to reach a desired LZ in obscured visibility conditions such as darkness, clouds, smoke, and other unforeseen situations. This allows missions to be carried out around the clock, even in adverse visibility conditions which would normally halt operations.

Acknowledgements

I would like to express my sincere appreciation to my thesis advisor, Dr. John Raquet for his endless support throughout my research. His insight and gentle nudging kept me from losing my mind completely.

I would also like to thank Don Smith for his support in putting together all the hardware components which went into this effort. His skill made it possible to bring this idea from my imagination into the physical world and actually have it function.

I would also like to thank my two sons and my daughter who always thought what I was working on was neat and even related it to a video game. Which is a great complement from a 13 year old.

And finally I would like to thank my girlfriend who actually took the time to read this entire document more than once. She was always able to pick out all the bad grammar. Thank you for keeping me grounded and giving me an out every once in a while to keep me for going completely crazy. Now its off to our next adventure!

Fernando Ontiveros

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation		Page
GPS	Global Position System	1
NVG	Night Vision Goggles	1
HUD	Heads-Up Display	1
HAHO	High Altitude High Open	1
CSAR	Combat Search And Recovery	2
PJ	Parachute Jumper	4
LZ	Landing Zone	4
HAHO	High Altitude High Open	5
LZ	Landing Zone	5
UTC	Coordinated Universal Time	9
TOA	Time Of Arrival	10
CDMA	Code Division Multiple Access	10
C/A	Coarse/Acquisition	10
HAHO	High Altitude High Open	13
INS	Inertial Navigation System	13
IMU	Inertial Measurement Unit	14
IR	Infrared Imaging	14
OPU	Overhead Projector Unit	16
CRT	Cathode Ray Tube	16
LCD	Liquid Crystal Display	17
LCoS	Liquid Crystal on Silicon	17
DMD	Digital Micro-mirrors Display	17
OLED	Organic Light-Emitting Diode	17
PFD	Primary Flight Display	17
NASA	National Aeronautics and Space Administration	18

Development of a Night Vision Goggle Heads-Up Display For Paratrooper Guidance

I. Introduction

The goal of this thesis is to develop a proof of concept for the incorporation of a Global Positioning System (GPS) display into a paratrooper's Night Vision Goggles (NVG) Heads-Up Display (HUD). The driving force behind the development of this research is to provide a small, light-weight, non-intrusive navigation system specifically designed for Special Operations and Airborne personnel. The project uses GPS and inertial technology to provide navigational data to a body worn computer system, which in turn generates the appropriate navigational display. The navigational display is a tunnel in the sky, presented to the operator via a color display generated within the Night Vision Goggles. This carries forward the work started in two previous thesis which addressed the same subject.

1.1 Background

United States military personnel have adapted jump tactics in order to utilize new technologies and increase the probability for mission success. For example, US paratroopers incorporate the use of High Altitude High Open (HAHO) jumps in order to decrease the chance of detection by having transportation aircraft standoff further distances. By allowing the paratroopers to open their parachutes at high altitudes, the paratroopers can travel greater distances to the landing zone. This prevents the aircraft from approaching the landing zone, thus reducing detection. The use of HAHO jumps, allowing the distant deployment of personnel, combined with the cover of darkness, greatly reduces the possibility of detection. The use of Night Vision Goggles has made it possible for personnel to travel the longer distances under the cover of darkness. Although, the use of NVG technology has given paratroopers the ability to see at night, it still does not provide them with any navigational data. HAHO jumps can cover long distances and without navigational data paratroopers must rely on visual cues to reach their desired target.

The first HAHO jump was conducted by USAF Colonel Joe Kittinger at an altitude of 19 miles above the earth's surface. The first use of a HAHO jump in a combat operation occurred during the Vietnam War [SEC07]. While HAHO jumps solved the problem of getting to a stranded airman, other problems have arisen. For instance, conducting a HAHO jump during daylight means an enemy on the ground could easily observe the paratroopers and landing location [THO05], and during inclement weather or smoke covered regions, such as the oil fires in Kuwait, the current NVG technology does not provide the necessary data.

Combat Search And Recovery (CSAR) efforts are also often adversely affected by bad weather or hostile environments, which prevent Special Operations personnel from jumping at safe altitudes. These factors can be greatly mitigated by providing the jumper with an accurate light-weight navigation option. Due to low visibility, CSAR operations often can not be undertaken because of the unsafe jump conditions. Since the cloud cover ceiling is so low, the rescue jumpers can not safely navigate or open their parachutes at a safe altitude, preventing them from reaching the downed or stranded personnel on the ground.

Current methods of presenting navigational information to paratroopers combine Global Positioning Systems (GPS) technology with chest worn and goggle attached displays [CRA08]. The current technology does not present the paratrooper with a full navigational picture. The display in figure 1.1 provides the following information:

- Number of satellites being tracked
- Ground track
- Selected LZ
- Ground track on compass (always points up)

- Heading to selected LZ
- Destination distance
- Altitude

While this system does provide the paratrooper with a great deal of information and can be used for daytime jumps, it lacks nighttime operational capabilities. Additionally, the current informational display does not present a display which is intuitive to jumpers. This lack of obvious navigational information prevents this from being an ideal solution for airborne personnel.

In addition, the current HUD navigation system does not utilize any head tracking technology. Instead the display is usually based on the GPS-derived velocity, not on the actual orientation(heading) of the parachute or the soldier's head. The head tracking technology allows the operator to look in directions which may not be the direction of the LZ. Due to the winds at various altitudes, the paratrooper may be forced to follow different headings in order to arrive at the various way points. The current HUD systems assume the operator is looking in the same direction as their heading, which can be erroneous. The paratrooper may be looking at a vector which is not the landing zone, but current technologies do not account for the head tracking needed to accomplish this task.

In order to provide the airborne personnel a greater chance of success, work was begun to develop a reliable navigational display system for airborne personnel. Work in this area was first begun in 2003 by Balaz, who developed a simulation running on a PC of the virtual tunnel landing system [BAL03]. The navigation reference was intended to serve as a Heads-Up Display integration into Night Vision Goggles used by paratroopers with High Altitude High Opening parachutes during inclement weather [BAL03]. The virtual tunnel in the sky was then integrated into a Night Vision Goggle's (NVGs) Heads Up Display operating on a portable laptop computer by Thompson [THO05].

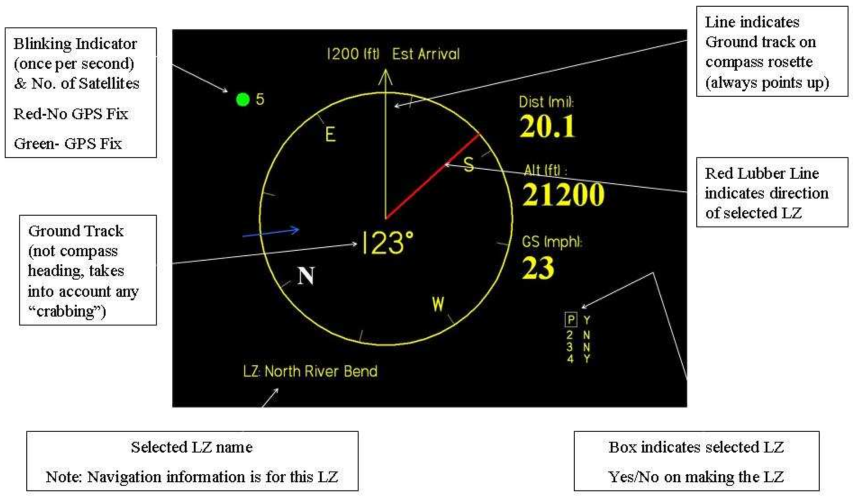

Blinking Indicator (once per second) & No. of Satellites
Red-No GPS Fix
Green- GPS Fix
Ground Track (not compass heading, takes into account any "crabbing")
Selected LZ name
Note: Navigation information is for this LZ
Line indicates Ground track on compass rosette (always points up)
Red Lubber Line indicates direction of selected LZ
Box indicates selected LZ
Yes/No on making the LZ
1200 (ft) Est Arrival
5
E
S
N
W
123°
Dist (mi):
20.1
Alt (ft):
21200
GS (mph):
23
P Y
2 N
3 N
4 Y
LZ: North River Bend

Figure 1.1: Current navigational displays [CRA08].

This thesis will take this project to the next step, providing a proof of concept, by integrating the software into an embedded computer system which can be carried by Special Operations Parachute Jumpers (PJ) through actual airborne parachute jumps. Additionally, a new display algorithm was developed which more efficiently calculates the desired trajectory.

This work combines several fields of research into a paratrooper's Night Vision Goggles (NVG) heads-up display (HUD) that provides an intuitive navigational display to the landing zone. Hardware was assembled and integrated into a small, light-weight package which could safely be carried by airborne personnel. In addition, software programs and algorithms to provide the realtime updates needed to guide personnel to the desired LZ were developed. The goal of this project is to provide paratroopers the tools to make landings on target regardless of Landing Zone (LZ) visibility or weather conditions and enable them to perform missions that were once impossible. By allowing specialized teams, such as CSAR personnel, to carry out their mission, lives will be saved and missions will be enhanced.

1.2 Research Focus

The primary focus of this research is to complete a proof of concept for the GPS display within a paratrooper's Night Vision Goggles. The path generating algorithm is based on work begun in the two previous theses. The project is composed of two distinct parts–the hardware and the software. The hardware focus of this project builds a system which is small, lightweight, and can be worn during an actual parachute jump. The software portion of the project is focused on writing a real-time program which is efficient and will provide an intuitive interface for the user.

1.3 Objectives

The primary objective of this research is to provide a proof-of-concept for the design and implementation of a NVG HUD system to convey navigational information a paratrooper needs in order to execute a High Altitude High Open (HAHO) jump and land at the designated Landing Zone(LZ).

The proof of concept is demonstrated in two steps. First, hardware selected and built to provide a small light-weight package. The hardware design takes several of the constraints into consideration, such as flight safety, weight and size. The overall system package must be capable of fitting into a paratrooper's ammo pouch, yet still provide the information in a clear and concise manner for the entire descent period. The second step developed the software algorithms that run efficiently on the smaller hardware configuration. This smaller configuration includes limited processing power and memory availability.

The software system integrates data from an integrated GPS/IMU (Inertial Measurement Unit), which measures real-time position and attitude of the jumpers helmet. The data is used to calculate a path for the paratrooper to follow in order to successfully land at the LZ. The path is depicted in an intuitive, three dimensional, graphical display, projected as a tunnel in the sky within the paratrooper's NVG HUD system.

1.4 Assumptions

Several assumptions have been made in order to limit the scope of this research to a manageable level. First, we assume that this navigational information will only be used in a parachuting environment. Although the system can be adapted to ground navigation, this is not the thrust of this research. Second, it is assumed the system will be used while the paratrooper is under an open parachute. If the system is employed prior to the parachute opening, the head tracking system will not function properly, due to the placement of the GPS antenna. The antenna has been placed in a location to optimize antenna signal reception while the operator is in a vertical position, under canopy. Third, it is assumed that wind information used while generating the path is reasonably accurate. Once the wind information is read into the system from the initialization file, all calculations are rendered from those values. If the time/wind values differ significantly from the entered wind information, the path generation will not be accurate and could therefore produce navigational paths which can not be achieved due to environmental factors. Fourth, it is assumed that the GPS/IMU information is correct. This equipment has been utilized by other systems and has proven to provide reliable information; therefore, once system checks have been passed, the information will no longer be verified and is assumed to be accurate.

1.5 Approach

We approach this thesis by developing hardware and software in a parallel. The hardware systems selected provide a mobile platform which is light-weight and powerful enough to handle the calculations which are necessary to provide the real-time navigational data. An additional hardware requirement is the need for operator safety, in which excessive weight must be kept off the head area, due to the extreme G-forces which operators are subject to during operations.

The software is the second portion of the system developed in parallel with the hardware. The kernel was built to operate the selected processing hardware. Items such as hardware memory limitations and processor speed are items which affected the software development. Software programs and algorithms were also developed to operate on the limited processor architecture.

The system uses a consistent process to generate the navigational display on the HUD. Information about the HAHO jump is collected in a parameter file prior to the jump, and is read into the system prior to departing the aircraft. The parameters collected include parachute characteristics, Landing Zone(LZ) coordinates, jumper's preferences, and wind information.

Once the information is collected, a path to the LZ is computed, based on the wind and parachute performance data. The path is displayed to the paratroopers as a series of boxes forming a virtual tunnel for them to fly through. The path begins at the initialization point (the position at which the system is initialized after canopy opening) and leads the jumpers to the Landing Zone.

The GPS and orientation data is generated from the GPS/IMU continuously as the paratrooper descends. The information is passed through the system and used to update the tunnel in the sky. GPS/IMU data provides current location and heading information.

The system was tested on the ground because of the risk to personnel and to build confidence in the overall reliability of the system. The first test evaluated the head tracking algorithms to ensure the operator was able to acquire and track the targeted landing zone in the correct physical location. The next portion of the ground test, tested the effect of the navigational aides to the user. This allowed the operator to exercise the system to ensure it did indeed provide an improvement over the current navigational methods.

Follow on evaluation of the system needs to be conducted by constructing a simulation which emulates the effects of winds while allowing inputs from the head tracking system to simulate the operational functionality. When confidence in the system reliability has been reached, only then should live testing be conducted. The live testing should only be conduced by qualified jump test individuals at locations such as Natick Soldier Evaluation Center.

1.6 Document Overview

The remainder of this thesis is divided into five chapters. Chapter two explores previous research conducted in the field of using a HUD as a reference in addition to other related research areas, such as head-tracking and parachute characteristics. Chapter three details the implementation of the hardware system. Chapter four describes the software methodology and description used in the system. Chapter five presents the various testing environments utilized. Chapter six presents the final test results and analysis of the data. Chapter seven summarizes the research and presents topics for further research and development related to this project.

II. GPS Display in Night Vision Goggle-Background

2.1 Navigation History

Navigation is the art and science of charting a course from point A to point B and staying on that course. This act of navigation can be as simple as driving to work or walking to a store, which can be accomplished though utilizing our eyes, common sense, and landmarks. However, there are many other instances which require more accurate knowledge of our position, intended course, or transit time to a desired destination. In these situations, the use of navigational aids assist us in correctly navigating to the destination. These navigational aids can be a simple clock to determine the velocity over a known distance, an odometer in the car to track distance travelled, or more sophisticated systems which transmit electronic signals, i.e. radionavigation aids [THO05].

The subject has a long and fascinating history, with ancient explorers crossing vast oceans guided only by the stars. They soon discovered stars couldn't be counted upon to always be visible. The technology of the twentieth century has now solved this problem nearly completely by placing artificial stars in the sky. The NAVSTAR Global Position System is the first of this new breed of global navigation satellite systems to become operational [MIS06].

2.2 GPS Overview

Presently, GPS is fully operational and meets the criteria established in the 1960s as an optimum positioning system. The system provides accurate, continuous, world-wide, three-dimensional position and velocity information to users with the appropriate receiving equipment. GPS also disseminates a form of Coordinated Universal Time (UTC). The satellite constellation nominally consists of 24 satellites arranged in 6 orbital planes with 4 satellites per plane, as shown in figure 2.1. A worldwide ground control/monitoring network monitors the health and status of the satellites. This network also uploads navigation and other data to the satellites. GPS can provide service to an unlimited number of users since user receivers operate

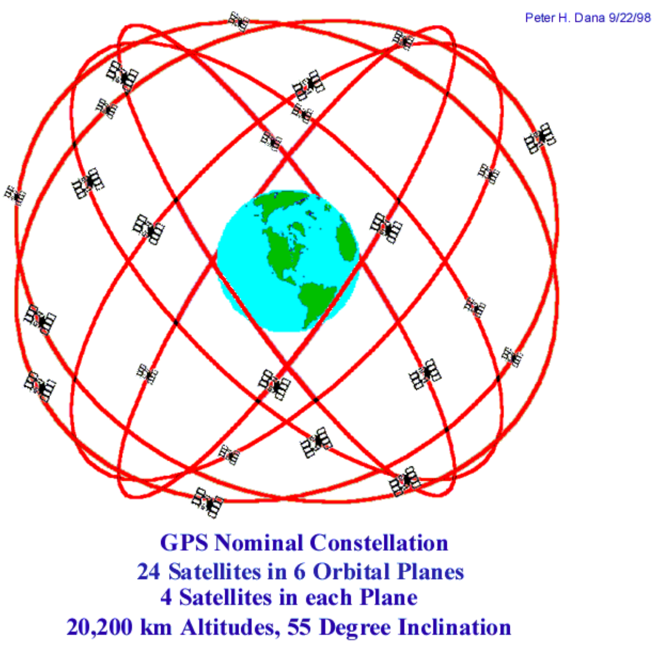

Peter H. Dana 9/22/98
GPS Nominal Constellation
24 Satellites in 6 Orbital Planes
4 Satellites in each Plane
20,200 km Altitudes, 55 Degree Inclination

Figure 2.1: View of the GPS constellation orbits [DAN98].

passively (i.e. receive only). The system utilizes the concept of one-way Time of Arrival (TOA) ranging. Satellite transmissions are referenced to highly accurate atomic frequency standards onboard the satellites, which are synchronized with a common GPS time base. The satellites broadcast ranging codes and navigation data on two frequencies (referred to as L1 and L2) using a technique called Code Division Multiple Access (CDMA). Each satellite transmits on these frequencies, but with different ranging codes than those employed by other satellites. Each satellite generates a short code referred to as the Coarse/Acquisition C/A code and a long code denoted as the Precision (P) code. Currently all satellites actually transmit an encrypted form of the P code called the Y-code.

The navigation data provides the means for the receiver to determine the location of the satellite at the time of signal transmission, whereas the ranging code enables the user's receiver to determine the transit time of the signal and thereby determine the satellite-to-user range. This technique requires that the user receiver also contain a clock. Utilizing this technique to measure the receivers's three-dimensional location requires that TOA ranging measurements be made to four satellites. If the receiver clock were synchronized with the satellite clocks, only three range measurements would be required. However, a crystal clock is usually employed in navigation receivers to minimize the cost, complexity, and size of the receiver. Thus, four measurements are required to determine user latitude, longitude, height, and receiver clock offset from internal system time. If either system time or height is accurately known, fewer than four satellites are required [KAP06].

2.3 Parachute

Leonardo da Vinci first sketched an idea for a device (a "tent roof") that would let someone down safely from high buildings, but he never moved this concept from the drawing board to production. The crude sketch depicted a large four-sided cloth covered framework resembling a pyramid in shape. Attached to each lower corner of the shape were four long suspension lines secured together at a confluence formed at the lower ends of the lines (see Figure 2.2). In use, da Vinci showed how someone would hang by hands from the line confluence and be lowered beneath the open shape to the ground [SEC06].

Since the time of da Vinci the parachute has continued to evolve to handle everything from military cargo and space vehicle recovery to use on racing vehicles [SEC06]. The parachute has allowed the military to develop safety systems for use in aircraft, and special operations which rely on dropping airborne personnel from miles above the earth and arriving safely at the desired location. The technology has proceeded from a solid fabric to a silk weave to the latest parafoil (see Figure 2.3) or ram chute technology used by military personnel.

The parafoil or ram-air parachute is a deformable airfoil that maintains its profile by trapping air between two rectangularly shaped membranes, sewn together at the trailing edge and sides, but open at the leading edge. Several ribs are sewn to the inside of the upper and lower surfaces, maintaining an airfoil cross section in the spanwise direction. The parafoil is a parachute-like device that can be steered. This system is designed to retard the vertical velocity and provide a relatively soft touchdown.

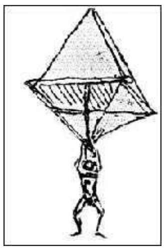


Figure 2.2: da Vinci's original parachute concept [SEC06].

Figure 2.3: Parafoil used to deliver cargo [SEC07].

The lifting parafoil has three advantages over the conventional type: 1) being able to reduce the dispersions associated with trajectories by using its maneuverability to glide to a predetermined point, 2) having the capability of being manually controlled to minimize landing area impact dispersions and, 3) by flairing, to reduce the impact shock at touchdown. The advantages of a conventional parachute are reduced weight and less complexity. If a parafoil is deployed at an altitude of approximately 6000 meters, typical performance characteristics provide a maneuvering circle of about 16 km in radius [SEC06].

The parafoil, or square parachute, is popular in sport parachuting but only sees limited use in the military, mainly among special operations units and demonstration teams. The majority of military parachutes are round in shape and have limited or no steering capability, which is important to large scale paratrooper operations. It is undesirable to have several hundred paratroopers independently steering their parachutes because of the risk of collision [SEC06].

2.4 High Altitude High Open

The system was developed to assist personnel during a High Altitude High Open (HAHO) descent. A HAHO jump is characterized by the personnel opening their chutes shortly after departing the aircraft. HAHO is generally used to airdrop personnel at high altitudes when aircraft can't fly above enemy skies without posing a threat to the aircraft and jumpers. In addition, HAHO parachute jumps are used to insert covert military personnel (generally special forces) into enemy territory in circumstances where the covert nature of an operation may be compromised by the loud noise of parachutes opening at low altitude.

In a typical HAHO exercise, the jumper will jump from the aircraft and deploy the parachute at a high altitude, 10 to 15 seconds after departing the aircraft (typically at approximately 27,000 feet (8300m)). The jumper will use a compass or GPS device for guidance while flying for 30 or more miles [SEC07]. The jumper must use way points and terrain features to navigate to his desired landing zone, and along the way, must correct his course for changes in wind speed and direction; making for a tricky navigation problem [SEC07].

2.5 Inertial Navigation System

Every object that is free to move in space has six "degrees of freedom". There are three linear degrees of freedom (x,y,z) that specify position and three rotational degrees of freedom (pitch, yaw, and roll) that specify orientation. If these six variables are known, then location and attitude are known. An Inertial Navigation System (INS) utilizes self contained accelerometers and gyroscopes to provide a navigation solution. The only difference between an INS and an IMU is that in an INS, the solution is calculated, but in an IMU, only the measurements are collected. Position and orientation can be acquired by integrating the accelerometers and rate gyros [UNI07].

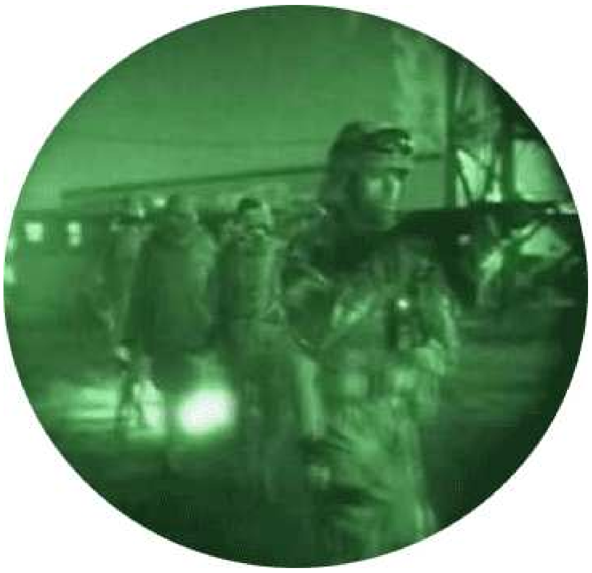


Figure 2.4: A typical NVG display seen by military personnel–Iraq 2004.

An Inertial Measurement Unit (IMU) is a set of six inertial sensors. An IMU is normally composed of three linear accelerometers and three rate gyros with a computational unit to conduct the position calculations based off the sensors. The problem with IMU's is the error which occurs from the sensors. Although the error can be small (in the millionths of g's from some accelerometers) it is nontrivial. This error begins to grow when it is integrated twice over long periods of time. The noise accumulates after integration into the solution, causing a slow drift which gradually gains speed. In order to correct for the drift, many IMU units utilize GPS data to null out the IMU drift error [UNI07].

2.6 System Hardware

2.6.1 Night Vision Goggle Technology. There are two major types of NVG technology which currently dominate the field: image intensification (light amplification) and thermal infrared (IR) imaging. Light amplification works as the name suggests, electronic signals (light detection) nearly undetectable by the human eye are passed through the amplification and altered into energy patterns (See Figure 2.4).

Just as faint sounds can be made louder, near total darkness can become visible. The NVG takes photons, and converts them into electrons. These electrons pass through a thin disk that is about the size of a quarter and contains over 10 million channels. As the electrons travel through and strike the walls of the channels, thousands of additional electrons are released. These multiplied electrons bounce off the phosphor screen, converting the electrons back into a greatly amplified number of photons, letting an operator see the nighttime view in extremely low light. The intensifier tube component is the workhorse of this type of night vision device. Intensifier tube performance is a very important factor when evaluating and selecting any night vision device.

The second prominent technology, thermal-imaging, looks at heat and not visible light. Unlike image-intensifiers, thermal-imaging is less affected by smoke or fog or other weather conditions as it is not dependant on visible light. Thermal-imaging has infrared detectors sensitive to the invisible infrared portion of the electromagnetic spectrum or heat. The image is typically seen on a color scale with the more expensive units or can be shown in shades of gray [DAV07].

2.6.2 Heads-Up Display. The Heads-Up Display (HUD) is a transparent display that presents data without obstructing the operator's view. Although the HUD was initially developed for military aviation (see Figure 2.5, HUDs are now used in commercial aircraft, automobiles, and other applications. There are two types of HUDs–fixed and helmet-mounted. Fixed HUDs require the user to look through a display element attached to the airframe or vehicle chassis. The system determines the image to be presented depending solely on the orientation of the vehicle. Commercial aircraft and automobiles usually incorporate a fixed HUD system. Helmet-mounted or head-mounted HUDs feature a securely-attached display element that moves with the orientation of the user's head. Such systems are often monocular and are used in the Apache attack helicopters and other fighter aircraft. Miniature HUDs are used to enhance combat readiness and situational awareness with unmatched sensor fusion capabilities for night or day operation. The HUD enables the operator to overlay real time alphanumeric or video information from thermal sights, personal computers, vehicle systems, or other key information systems onto night vision view [ANS07].

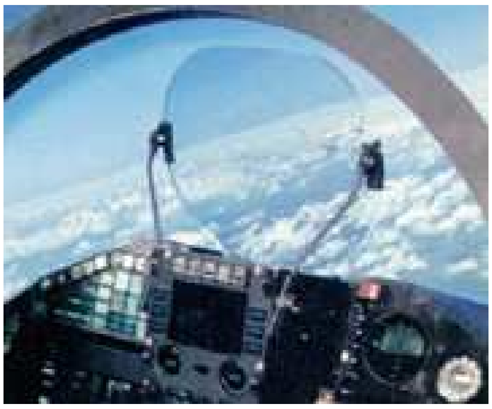


Figure 2.5: F-22 HUD example; standard HUD currently utilized in military aircraft [QSA08].

A typical HUD in civil aircraft contains three primary components: a computer, an Overhead Projector Unit (OPU) and a combiner. The computer usually is located with the other avionics equipment and provides the interface between the HUD and the aircraft systems to be displayed. Flight data is received from the inertial reference system, flight management system, and other flight guidance systems, and then it is processed into a form compatible with the Overhead Projector Unit. The CPU takes this data and projects it onto the combiner. The combiner is usually made of glass with a special coating that reflects the monochromatic (monochromatic light is described by only one frequency, such as a laser [ELE08]) light from the OPU, while allowing all other wavelengths of light to pass through, creating a superimposed image. Tactical military aircraft usually rely on a projection unit incorporated into the combiner [SPI07] (See Figure 2.6).

Traditionally, the source for the projected image has been a Cathode Ray Tube (CRT). This is the same technology traditionally used in older televisions and monitors, but micro-display imaging technologies are now being introduced which are opening up new applications for HUD technology. Currently, micro-display technologies

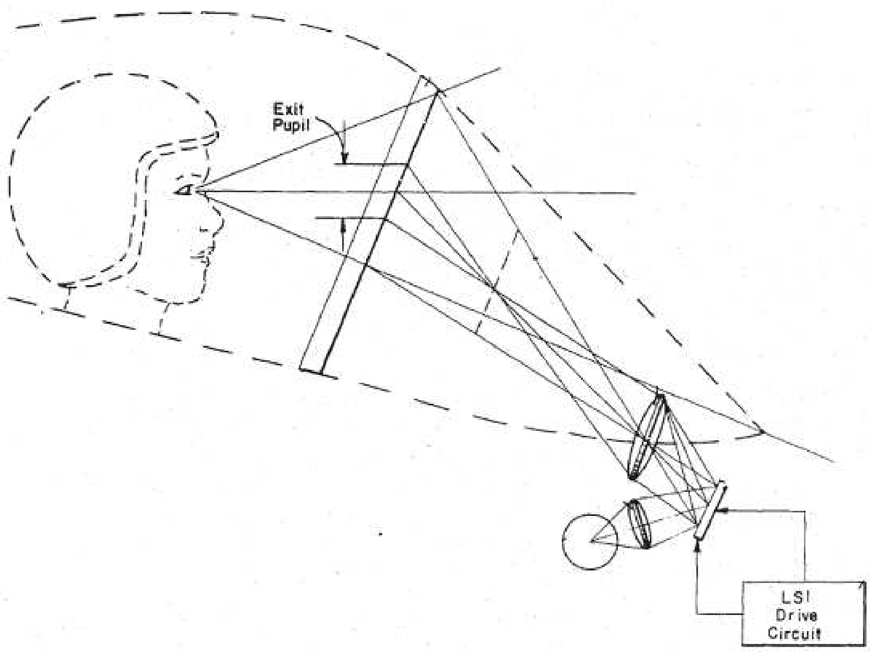


Figure 2.6: Standard military aircraft HUD operation [ANS07].

that have been demonstrated include Liquid Crystal Display (LCD), Liquid Crystal on Silicon (LCoS), Digital Micro-mirrors Display (DMD), and Organic Light-Emitting Diode (OLED) [TEK06] (See Figure 2.7). HUD systems that project information directly onto the wearer's retina with a low-powered laser (virtual retinal display) are also in the experimentation stage [ANS07].

2.6.3 Tunnel in The Sky Navigation Displays. MIL-STD-1787 defines the standard military Head Up Display. The primary focus of the MIL-STD-1787 HUD is to convey flight information, such as the pitch, altitude, and airspeed of the aircraft. However, the HUD also displays additional information, such as the bearing, to the next way-point, course indicator, and vertical deviation indicator. For USAF purposes, the MIL-STD-1787 HUD conveys sufficient information to act as a Primary Flight Display (PFD) [THO05]. While the PFD is useful for flight information, it does not display sufficient information to produce a navigational display. The MIL-STD-1787 HUD is based on gauges and instruments that have been present in the cockpit since World War II [KRA98].

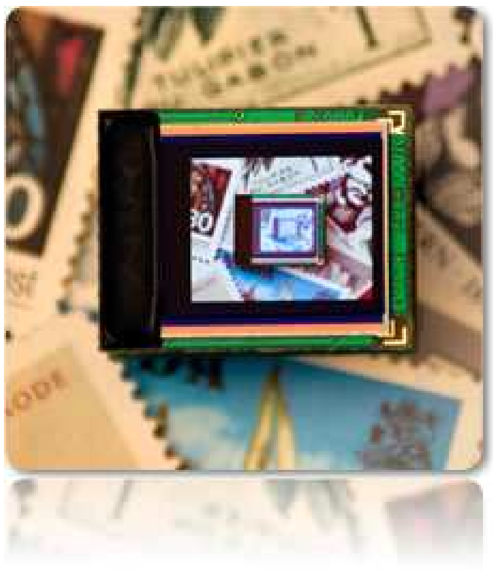


Figure 2.7: Size of OLED compared to postage stamps. OLED is utilized by STS in their AN/PVS-21 LPNVG HUD system [STS07].

The pathway in the sky originated because of the increased complexity of flying and navigating versus the information display. Commercial airliners often fly in situations, such as landing patterns, which require them to follow precise flight profiles in order to avoid collisions with other aircraft. Because the MIL-STD-1787 HUD is inadequate for this type of flight, other methods to display navigational information have been developed. One such technique is the pathway in the sky method. During the mid-1990's, the National Aeronautics and Space Administration (NASA) developed a Synthetic Vision System (SVS) to preset real world information to aircraft/spacecraft pilots. This had the advantage of eliminating cockpit windows and gauges from the aircraft/spacecraft design [ANS07]. Once the initial concept had been implemented, further research progressed on the most effective display for pilots. Research conducted on several different approaches began to emerge. For example, displays ranged from two-dimensional grid square displays to more compelling three-dimensional computer graphics. With the increase in computing power and computer graphics, the navigational displays could display many different styles of pathways in the sky.

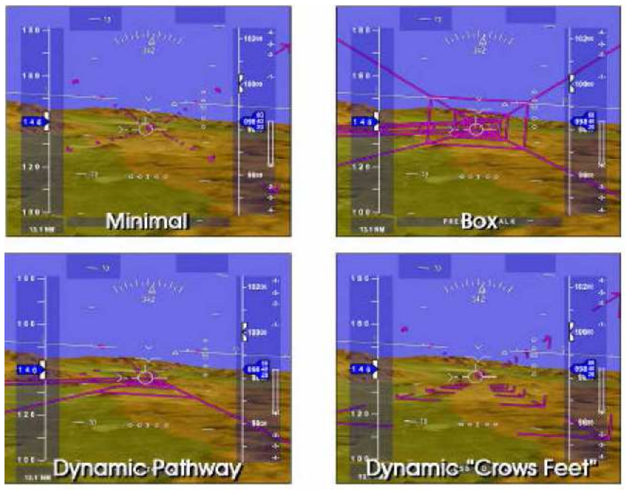


Figure 2.8: Graphic view of the four HUD designed [PAR03].

Research conducted by [SNO99] found that pilots using the pathway in the sky HUD were able to land with the same accuracy and maintain a commanded airspeed equally well, regardless of visibility conditions. Pilots preferred the pathway in the sky technique over the standard military HUD, as well, claiming that the pathway increased situational awareness, even when the ground was not visible [THO05]. Several methodologies for building the tunnels have been developed, ranging from the minimalist to the complex. Some configurations use a flat series of rectangles, creating a pathway resembling a sidewalk suspended in the air. These configurations create a flat path and are known as pathways in the sky [KRA98]. Other configurations use rectangles to define a tunnel around the path that should be taken by the pilot. These configurations are known as tunnels in the sky [SNO99]. All four displays are illustrated in figure 2.8.

2.6.4 STS Model AN and PVS-21 NVG. The platform used to display the GPS navigational information to the operator is the Sensor Technologies Systems (STS) Low Profile Night Vision Goggle model AN/PVS-21 (See Figures 2.9 and 2.10). The AN/PVS-21 is a Low Profile Night Vision Goggle (LPNVG); this system was selected because of its integrated SVGA HUD and third generation optics. The AN/PVS-21 utilizes OLED technology produced by eMagin, producing a 12.78 x 9mm viewing area in a 0.61" package size. The AN/PVS-21 LPNVG brings the optics closer to the wearer's face enhancing depth perception by using a unique "see-through" beam combiner system capitalizing on both intensifier (night vision enhanced) and unintensified (see-through capability for day use of HUD display) vision. This see-through technology permits operation from extreme low-light environments to bright light conditions. The miniaturized HUD option for the AN/PVS-21 can currently display map overlay, building schematics or any other PC display while leaving an unobstructed forward view. The HUD has 320 x 240 resolution and is horizontally and vertically adjustable [DAV07]. The AN/PVS-21 NVG has been certified for static line, HALO and HAHO parachute jumps. The NVG tubes are the latest Generation 3 tubes with an Objective focus from 25 cm to infinity and the entire AN/PVS-21 system weighs 1.68 lbs with a 7.85 in-lb torque (jump weight acting on the paratroopers head) [STS07].

2.6.5 Processing Board X-board GP8. The X-board GP8 is the second ARM-compatible XScale-based module in Kontron's X-board product family of small embedded form factor modules (See Figure 2.11). It is based on the Intel 80219 General Purposed PCI processor (400 MHz/600 MHz) and utilizes the Silicon Motion SM501 chip set. It has 128 MB of onboard Dynamic Random Access Memory (DRAM) and 8 MB of onboard dedicated graphic memory. The SM501 chip set enables accelerated graphics and video to support the AN/PVS-21 HUD display. In addition to the video support, the Kontron X-board GP8 module was selected because of its compact size and robust interface capabilities. The entire board is the size of credit card and does not require external cooling, adding to its low power consumption. In addition to the video support described above, the GP8 board also supports multiple interfaces such as Ethernet, Universal Serial Bus (USB) 2.0 and audio, which are required for reading data into the system [KON06].

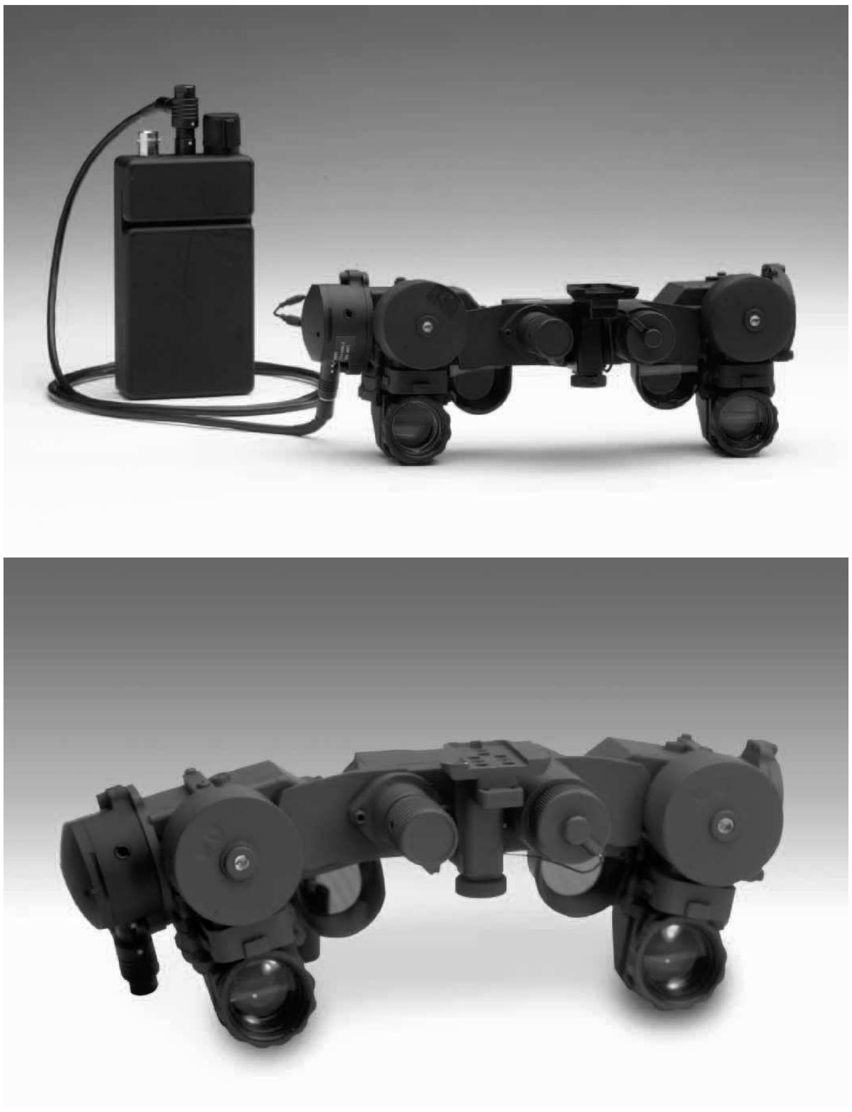


Figure 2.9: TOP: STS AN/PVS-21 Low Profile NVG (LPNVG) with battery pack connected to HUD. BOTTOM: AN/PVS-21 without battery pack connected [STS07].

Figure 2.10: Two STS Night Vision Goggle models mounted on desert helmets. LEFT: AN/PV2-21 LPNVG (used for this project). RIGHT: AN/PVS-15 [STS07].

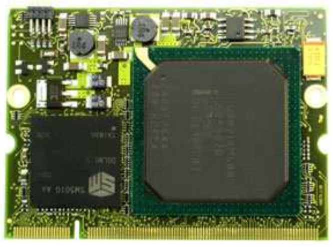


Figure 2.11: Kontron's GP8 X-Board. Large chip on dominating right side of board is the Intel 80219 CPU. Smaller chip on left is the SM501 Graphics chip [KON06].

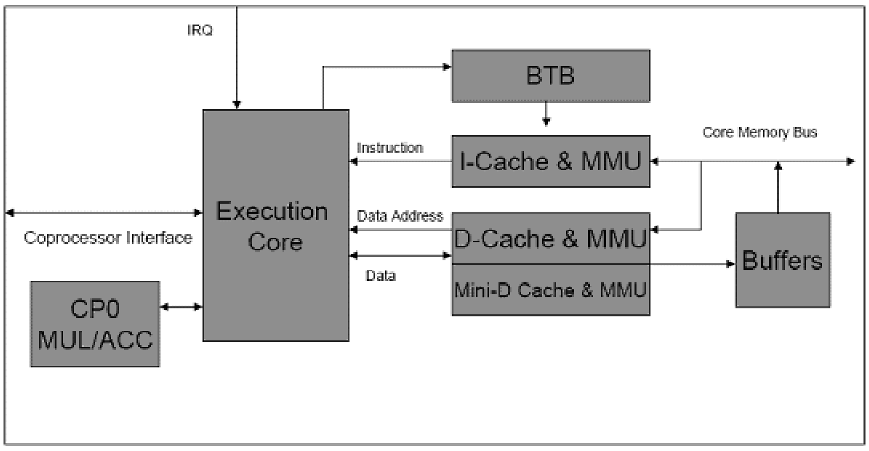


Figure 2.12: Intel X-Scale processor hardware overview.

Intel XScale microarchitecture is based on a new core which is compliant with ARM version 5TE (See Figure 2.12. The microarchitecture surrounds the core with instruction and data memory management units; instruction, data, and mini-data caches; write, fill, pend, and branch target buffers; power management, performance monitoring, and debug units; coprocessor interface; 32k caches; Memory Management Units (MMU), Branch Target Buffer (BTB), Multiplier-ACcumulator (MAC) coprocessor; and core memory bus [INT07a].

2.6.6 Head Tracking. In order to orient the navigational display with the operator's view point, the Microbotics' MIDG II was selected (See Figure 3.7). The MIDG II is an Inertial Navigation System (INS) with GPS contained in a small package. The MIDG II is a GPS aided inertial navigation system for use in applications

Figure 2.13: Microbotics' MIDG 2 provides Inertial Navigation System (INS) and GPS data [MIC07].

requiring altitude, position, velocity, acceleration, and angular rates for navigation or control. An internal GPS receiver measures position and velocity and passes it to the data fusion processor to be combined with the inertial data to generate an optimal solution. An internal three-axis magnetometer provides a magnetic heading reference when needed [MIC07].

2.7 Software

Ever since the invention of Charles Babbage's difference engine in 1822, computers have required a means of instructing them to perform a specific task. This means is known as a programming language. Computer languages were first composed of a series of steps to wire a particular program; these morphed into a series of steps keyed into the computer and then executed; later these languages acquired advanced features such as logical branching and object orientation. The computer languages of the last fifty years have come in two stages, the first major languages and the second major languages, which are in use today [FER04].

2.8 Programming Language

There are several programming languages that provide the operating capabilities for this program. Languages like JAVA and C# provide an object−oriented structure which allow the programmer to write object−oriented code, making the program easier to maintain and expand. The main drawback of these two languages is the need for a virtual machine running on the processor. The need for this virtual machine consumes valuable memory and processing resources, which are extremely limited in an embedded system. Other languages, such as C, provide the ability to execute in a limited processing environment and also provide the programmer with direct access to system resources, such as low-level memory. This is both a benefit and a hazard since the program can over-write critical sections of memory if great care is not taken by the programmer. An additional drawback to the C programming language is the inability to create object structures. This forces the programmer to write in a functional programming structure which has proven to be more difficult to maintain and debug. Another alternative is C++, which is considered a superclass of the C programming language.

C++ is an "object oriented" programming (OOP) language created by Bjarne Stroustrup and released in 1985. It implements "data abstraction" using a concept called "classes", along with other features to allow object-oriented programming. Parts of the C++ program are easily reusable and extensible; existing code is easily modifiable without actually having to change the code. C++ adds a concept called "operator overloading" not seen in the earlier OOP languages and it makes the creation of libraries much cleaner [CHE07].

C++ maintains aspects of the C programming language, yet has features which simplify memory management. Additionally, C++ contains features allowing low-level access to memory, along with high-level features such as object oriented programming capabilities. C++ could be considered a superset of C, which will run in C++ compilers. C uses structured programming concepts and techniques while C++ uses object oriented programming and classes which focus on data [CHE07].

2.9 Cross Compiling

A compiler is a program that turns source code into executable code. Like all programs, a compiler runs on a specific type of computer, and the new programs it outputs also run on a specific type of computer. The computer the compiler runs on is called the host and the computer the new programs run on is called the target. When the host and target are the same type of machine, the compiler is a native compiler. When the host and target are different, the compiler is a cross compiler.

In theory, a PC user who wanted to build programs for some device could acquire the appropriate target hardware (or emulator), boot a Linux distribution on that system, and compile natively within that environment. While this is a valid approach, it has a few prominent shortcomings for items such as a router or iPod. Drawbacks such as speed, capability, availability, and flexibility [LIN08] make development on these systems extremely taxing and time consuming. These factors force many programs to be written in different environments such as the X86 hardware, which has its own drawbacks. Word size, endianness, alignment, and default signedness are only a few of the obstacles which prevent programs from running on target systems which are not X86. In order to solve these issues, the program must be cross compiled or built with the appropriate word size, endianness, alignment and specifics of that particular target architecture. The process of cross compiling is most prominent when programming for embedded systems, whose limited memory, and processing capabilities make it infeasible to write, test, and debug on the target architecture.

2.10 Kernel

The kernel is a program that constitutes the central core of a computer operating system. It has complete control over everything that occurs in the system. A kernel can be contrasted with a shell (such as bash, csh or ksh in Unix-like operating systems), which is the outermost part of an operating system and a program that interacts with user commands. The kernel itself does not interact directly with the user, but rather interacts with the shell and other programs as well as with the hardware devices on the system, including the processor, memory and disk drives. The kernel is the first part of the operating system to load into memory during booting, and it remains there for the entire duration of the computer session because its services are required continuously. Thus it is important for it to be as small as possible while still providing all the essential services needed by the other parts of the operating system and by the various application programs [PRO05].

Because of its critical nature, the kernel code is usually loaded into a protected area of memory, which prevents it from being overwritten by other, less frequently used parts of the operating system or by application programs. The kernel performs its tasks, such as executing processes and handling interrupts, in kernel space, whereas everything a user normally does, such as writing text in a text editor or running programs in a GUI, is done in user space. This separation is made in order to prevent user data and kernel data from interfering with each other and thereby diminishing performance or causing the system to become unstable and possibly crashing.

2.10.1 Operating System.

2.10.1.1 Windows CE. Windows CE is the smallest Microsoft Windows operating system designed from the ground up to be a small Read Only Memory (ROM) based operating system with the Win32 subset Application Programming Interface (API). Windows CE extends the Windows API into the markets and machines that can not support the larger footprints of the Windows XP kernel. The Windows NT/2000/XP line of operating systems is written for the enterprise sector. It sacrifices compatibility and size to achieve its high level of reliability and robustness. Windows XP home Edition is a version of Windows XP built for the home user that does strive for compatibility, but this is secondary to its primary goal of stability. Windows CE isn't backward compatible with MS-DOS or Windows. Nor is it an all-powerful operating system designed for enterprise computing. Instead, Windows CE is a lightweight, multi-threaded operating system with a graphical user interface. Its strength lies in its small size, its Win32 subset API and its multi-platform support [BOL03].

Windows CE's greatest advantage is easily its similarity to other members of the Microsoft Windows family of operating systems. Many professional software developers have experience developing applications to the Windows API using Microsoft tools such as Visual C++ and Visual Basic. The ability to leverage those skills in the future is simply too large a factor to discount Windows CE. In addition, Microsoft's Windows CE is currently used as the foundation for the initial version of the .NET Compact Framework, a version of the .NET runtime for mobile and embedded devices. The Compact Framework provides the same powerful .NET runtime environment with a smaller class library so that it fits in small battery-powered devices.

2.10.1.2 Linux. Linux is a Unix-like operating system designed to provide personal computer users a free or very low-cost operating system comparable to traditional and more expensive Unix systems. Linux has a reputation as a very efficient and fast-performing system. Linux's kernel was developed by Linus Torvalds at the University of Helsinki in Finland [PRO07]. To complete the operating system, Torvalds and other team members made use of system components developed by members of the Free Software Foundation for the GNU Project. Linux is a complete operating system, including a graphical user interface, an X Window System, TCP/IP, and other components usually found in a comprehensive Unix system. Although copyrights are held by various creators of Linux's components, Linux is distributed using the Free Software Foundation's copyleft stipulations, which means any modified version that is redistributed must in turn be freely available [PRO05].

Unlike Windows and other proprietary systems, Linux is publicly open and extendible by contributors. Because it conforms to the Portable Operating System Interface standard user and programming interfaces, developers can write programs that can be ported to other operating systems. Linux comes in versions for all the major microprocessor platforms including the Intel, PowerPC, Sparc, ARM, and Alpha platforms.

2.11 Alternative Technologies

2.11.1 NVG Displays. There are numerous other projects currently being developed and fielded in order to provide the paratrooper and the Special Operations community with clear, easy to read navigation data. The Air Force Research Laboratory (AFRL) is doing extensive work in developing small body worn computers which will have integrated GPS and provide the operator a small portable computer [DAN08]. Although this does provide a GPS solution, this option does not provide the navigational data into a hands free system, such as an operator's NVGs. This will not work for personnel as they perform their free fall operations or even ground operations which require their full attention, such as combat missions. An alternative, which is also being developed by AFRL, is a Night-Vision Monocle which displays compass heading information. This is much easier for the operator to utilize, but the system still does not display GPS information such as latitude, longitude and altitude [DAN08].

2.11.2 Head Trackers. The head tracking arena has a number of new directions and initiatives currently being undertaken and sponsored by the Air Force. A number of these have being developed for inclusion in aircraft such as the F-35 Joint Strike Fighter. By developing the head trackers for cockpits, the system can determine where the pilot is looking in order to correctly ordinate threat warning tones. This will allow the pilot to react and search in the direction of the threat and not force the pilot to check the instruments prior to searching for the threat. One such technology which is currently under development consists of two microrings which contain fluid and sensors around the ring. As the pitch, yaw and roll change, the change in fluid orientation within the ring is registered by the sensors [DAN08]. Although this type of technology is promising because of its size and weight, the technology is still in the development stage and will not be available for use with this project.

2.11.3 Software - Optimized Algorithm. A change in the software approach to this project was also explored with the possibility of creating an optimized software algorithm based on advanced software methodologies. This approach relates the flight path problem to the 3-Dimensional routing problem utilized by the artificial intelligence field. Extensive work has been conducted in the 3-D routing arena in respect to Motion Planning (MP) for robotics. This is one of the most important areas of robotic research since the complexity of the motion-planning problem has hindered the development of practical algorithms. Until recently, robots were primarily employed for carrying out programmed repetitious task. Methodologies and algorithms for autonomous functioning were examined, but their implementation was hindered by the slow computing hardware. With the rapid advances in semiconductor and computing technology, it has become feasible to build robots that can function at reasonable speeds. The board classification of gross motion planning is what will be related to this problem. Gross motion planning is concerned with the problem involving free space much wider than the objects' sizes. Instead of a robot determining a 3-D route, the GPS system will determine the optimum route based on the current state. The algorithm will need to be effective and efficient in order to allow for the near realtime corrections at each stage [HWA92].

Certain constraints exist on the paratrooper, which do not directly relate to the robotic model, but still need to be taken into consideration. The primary constraint affecting the algorithm design is the various wind speeds at different altitudes during the descent. In addition to the wind speed, the overall paratrooper's velocity would also play a role in the characteristics of the proposed flight path. Although this approach was explored and defined, in order to produce the proof of concept with a working system, the software optimization option was not implemented. Instead of the 3-Dimensional approach, the software will instead route the most direct route to the landing zone, taking into consideration wind speed and operator velocity.

2.12 Previous Research

In 2003, Balaz began research into the area of providing airborne personnel with a reliable all weather navigation reference system. His primary focus was the design, implementation and evaluation of an intuitive three-dimensional display to serve as a primary navigation reference for paratroopers. A significant portion of his research involved the development of a software system to compute and display the optimal path to take a paratrooper from the release point to target. Although the research did not provide a fully operational version of a primary three-dimensional heads-up primacy navigation reference display for paratroopers using HAHO parachute techniques, it did provide the groundwork for further application in terms of software engineering and visual display [BAL03]

In 2005, Thompson continued the work begun by Balaz two years earlier. Thompson's work focused research to extend the design, implementation and evaluation of a prototype HUD displayed on NVG, using data from GPS and an Inertial Measurement Unit to enable a paratrooper to navigate in zero-visibility situations. A significant portion of his research was dedicated to developing a graphical HUD which had a low computing demand, enabling it to run on low-powered, easily portable computers and on how to compute a path to deliver a paratrooper from the release point to the landing zone. Another key point of research was to evaluate the effectiveness of the graphical HUD against conventional navigational cues [THO05]. The first of the two objectives; designing and implementing a HUD as a primary navigational reference for paratroopers was successfully achieved. A path generating algorithm was created and implemented by utilizing a third party graphics library. The second objective of testing the HUD system failed to meet its full objectives due to the lack of a fully built system. The system which was tested lacked key elements such as head tracking which proved to be crucial. Testing of the system actually proved a GPS receiver alone provided better navigational capabilities than the HUD system without head tracking.

2.13 Summary

This thesis builds upon work performed in two previous thesis in order to develop a proof of concept for this project. The three distinct areas of research are the navigation, hardware and software. The navigational aspect of the project will consist of utilizing GPS technology combined with head-tracking data to provide the positional, velocity, altitude and head orientation information. This information will then be processed by the processor worn by the operator in order to generate the correct flight path data along with the display information needed to produce the navigational display or tunnel in the sky. The software developed for larger footprint processors is ported into a the compact footprint needed in the Windows CE environment. GPS provides the navigation needed for the processor to display the tunnel in the sky navigational display to the operator.

III. Hardware Description

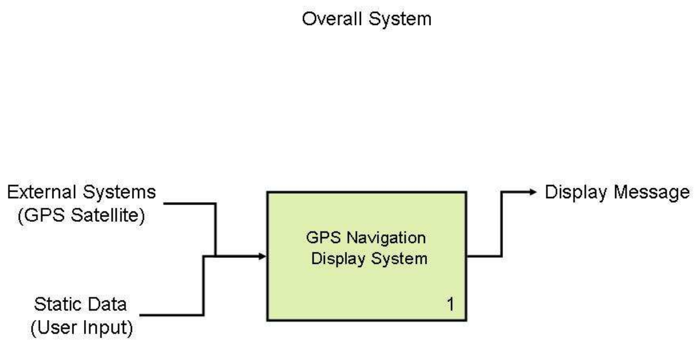

Overall System
External Systems (GPS Satellite)
Static Data (User Input)
GPS Navigation Display System
1
Display Message

Figure 3.1: High level view of the GPS Navigational Display System.

This chapter discusses the design and implementation of the hardware functionality for this system. The system is described by decomposing the entire system into smaller more atomic functional units. For example, to begin, the overall system is encapsulated in a single operational component, "GPS Navigation Display System", but as the chapter proceeds the system is decomposed into specific components. The software is approached in a similar manner in the following chapter.

3.1 System Description

The system as a whole is composed of three major hardware components; external inputs (external systems/sensors/user inputs), the body-worn computer, and the display system. The operational flow of the system consists of the sensors (GPS antenna and Head Tracker) receiving and forwarding the data to the body worn computer. The data is then parsed, decoded, and fed into the path algorithm to generate the navigation message which can then be sent to the NVG display system (See Figure 3.1).

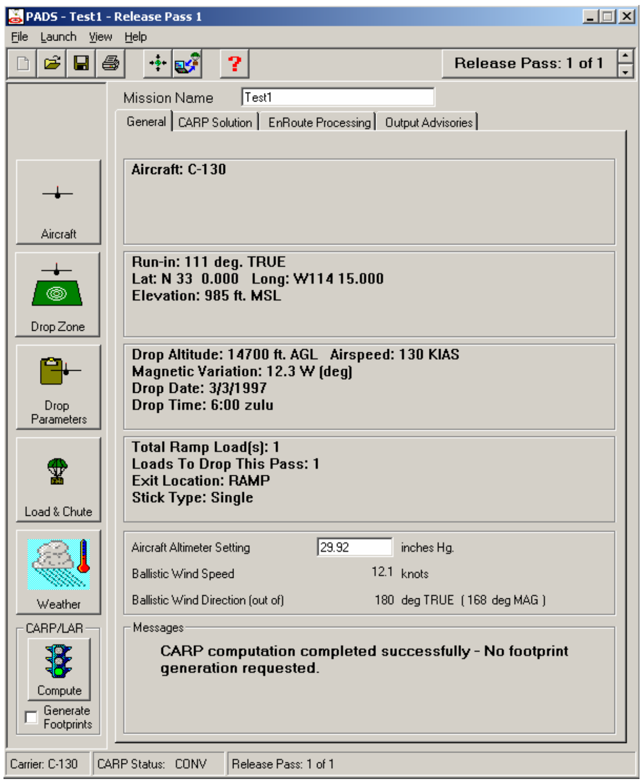
3.2 External Systems

The system interacts with external sources to achieve its overall goal of guiding the paratroopers to their final destination. The two external systems are 1) external jump parameters and 2) Global Positioning System. The first is needed to provide data related to the jumper's present environment which would affect the jump characteristics. The second external system provides the system with accurate position and velocity throughout the descent. The final portion of the external system is the sensors needed to receive the GPS data and accurately track any head orientation change.

3.2.1 External Jump Parameters.

3.2.1.1 Static Data. The first set of external inputs needed by the system is the static data input by the operator. This data is key to developing an accurate and achievable path from the release point to the landing zone. The operator is required to manually input certain variables, which will be used to determine the path generation. The four parameters needed by the system are the wind data at the various levels, the parachute profile, the release point, and the landing zone. This data must be input prior to departing the aircraft to generate the proper path generation solution. Since this data is completely autonomous to the system, the method must follow strict formatting rules. The most efficient and effective way to accomplish this formatting accuracy is to utilize the JPADS mission planning system. The JPADS system provides the user with a graphical interface for the entry of the environmental static data (See Figure 3.2). Although the JPADS system is the preferred input system, the user can also modify a stand-alone text file, but must once again follow the formatting rules.

3.2.2 User Input.



Mission Name Test1

Aircraft: C-130

Run-in: 111 deg. TRUE
Lat: N 33 0.000 Long: W114 15.000
Elevation: 985 ft. MSL

Drop Altitude: 14700 ft. AGL Airspeed: 130 KIAS
Magnetic Variation: 12.3 W (deg)
Drop Date: 3/3/1997
Drop Time: 6:00 zulu

Total Ramp Load(s): 1
Loads To Drop This Pass: 1
Exit Location: RAMP
Stick Type: Single

Aircraft Altimeter Setting 29.92 inches Hg.
Ballistic Wind Speed 12.1 knots
Ballistic Wind Direction (out of) 180 deg TRUE (168 deg MAG)

Messages
CARP computation completed successfully - No footprint generation requested.

Carrier: C-130 CARP Status: CONV Release Pass: 1 of 1

Figure 3.2: JPADS Main input screen.

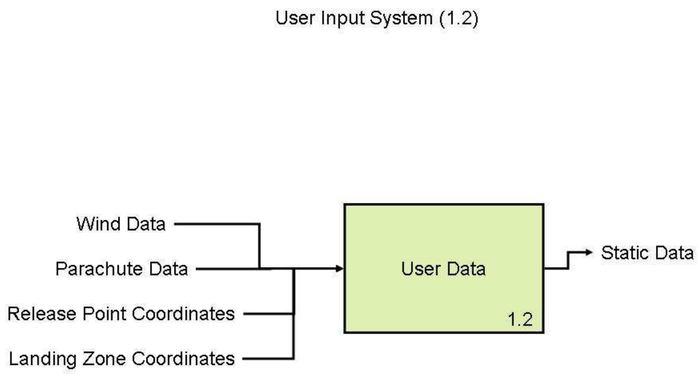
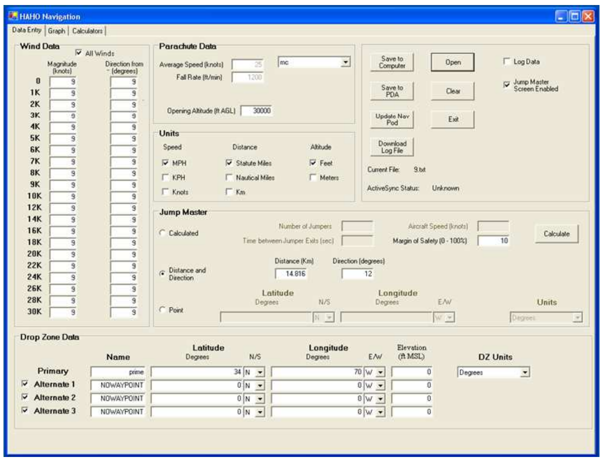


Figure 3.3: Required static user inputs, required prior to aircraft departure.



Wind Data ☑ All Winds

	Magnitude (knots)	Direction from (degrees)
0	9	9
1K	9	9
2K	9	9
3K	9	9
4K	9	9
5K	9	9
6K	9	9
7K	9	9
8K	9	9
9K	9	9
10K	9	9
12K	9	9
14K	9	9
16K	9	9
18K	9	9
20K	9	9
22K	9	9
24K	9	9
26K	9	9
28K	9	9
30K	9	9

Parachute Data: mc
Average Speed (knots): 25
Fall Rate (ft/min): 1200
Opening Altitude (ft AGL): 30000

Units
Speed: ☑ MPH ☐ KPH ☐ Knots
Distance: ☑ Statute Miles ☐ Nautical Miles ☐ Km
Altitude: ☑ Feet ☐ Meters

☐ Log Data
☑ Jump Master Screen Enabled
Current File: 9.txt
ActiveSync Status: Unknown

Jump Master
○ Calculated — Number of Jumpers: ; Time between Jumper Exits (sec): ; Aircraft Speed (knots): ; Margin of Safety (0 - 100%): 10
● Distance and Direction — Distance (Km): 14.816; Direction (degrees): 12
○ Point — Latitude Degrees: ; N/S: N; Longitude Degrees: ; E/W: W; Units: Degrees

Drop Zone Data

	Name	Latitude Degrees	N/S	Longitude Degrees	E/W	Elevation (ft MSL)
Primary	prime	34	N	70	W	0
☑ Alternate 1	NOWAYPOINT	0	N	0	W	0
☑ Alternate 2	NOWAYPOINT	0	N	0	W	0
☑ Alternate 3	NOWAYPOINT	0	N	0	W	0

DZ Units: Degrees

Figure 3.4: HAHO Mission Planner Main input screen.

3.2.2.1 Wind Data. Wind data is required to assess the possible travel distance an operator can cover given the parachute profile and the wind characteristics at various altitudes. For example, if an airfoil can travel 2 miles within a 100 foot altitude drop in zero wind, the system must account for winds reducing or increasing the travel distance of a given parachute. The wind data must be acquired by external means. Systems such as JPADS (See Figure 3.2 or the HAHO Mission Planner (See Figure 4.1) automatically gather information from air dropped transponders. A second approach is to have personnel manually note the wind characteristics as the aircraft travels up through the various altitudes.

3.2.2.2 Parachute Data. The second data element the operator needs to input is the parachute profile. This is needed to provide an achievable path given the parachute performance characteristics. For example, a pilot's ejection chute can not travel as far and is not as maneuverable as an airfoil designed for special operations personnel. Given the different characteristics, the user will have to input the correct parachute model utilized for the jump.

3.2.2.3 Release Point and Landing Zone coordinates. The third and fourth data elements needed by the system are the Release Point (RP) and the Landing Zone (LZ) coordinates. The operator will need to enter the latitude and longitude

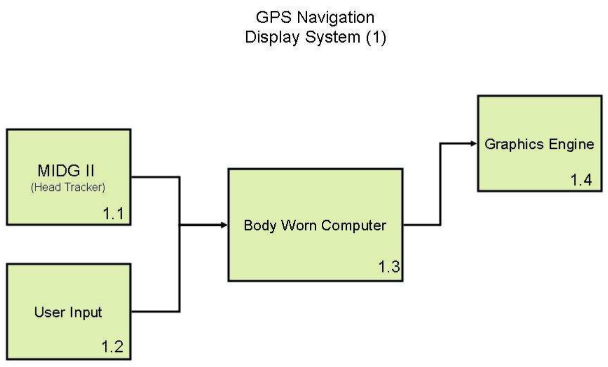


Figure 3.5: Components making up the GPS Navigation Display System.

and the altitude for the LZ. In addition, to have the system generate a valid path, the point at which the system will begin tracking is needed to generate the initial navigation solution. The information must be input prior to departing the aircraft.

3.2.3 GPS Navigation Display System. The second external system needed is the Global Positioning System, which provides a signal to be utilized for developing position and velocity solutions. The GPS signal is also utilized by the MIDG for updating the INS solution to improve attitude performance. The MIDG, like all IMU's, is suspectable to drift due to small errors in its accelerometers and gyroscopes. These small errors are introduced into the solution and are propagated continuously, developing a greater and greater amount of drift. The MIDG utilizes the GPS solution to null the effects of the drift and provide an accurate solution.

The external data noted above is passed into the system, which utilizes algorithms described in Chapter 4, to generate a navigational message. The navigational message is processed and sent to the external display system. The display can connect to any standard CRT/video monitor. The system outputs the graphical display utilizing 800 x 600 pixel SVGA resolution and can therefore be connected to any monitor system capable of displaying this resolution. The system utilizes the Sensor

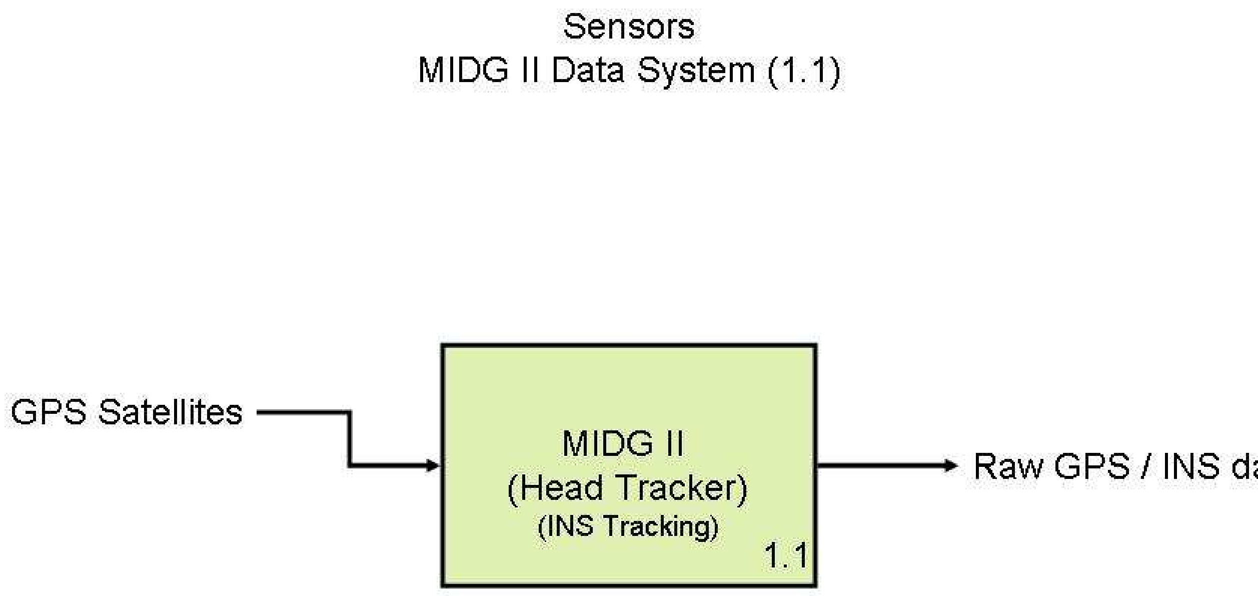


Figure 3.6: The MIDG II subsystem which provides sensor data.

Technology Systems (STS) HUD system to display the navigational information via the NVG's. The STS HUD is designed to utilize the SVGA output in order to provide the operator with a clear site picture. The interconnection between the MIDG and the remainder of the system in shown in Figure 3.5.

3.2.4 MIDG II - Head Tracker. The head tracking unit consists of the MIDG II, which provides the system with its attitude data. The MIDG is an Inertial Navigation System (INS) with Global Positioning System (GPS) within a single unit. All data messages are timestamped with GPS time, and a one pulse per second signal is used for synchronization with the body worn computer system. The MIDG II can operate in Inertial Measurement Unit (IMU) only or full INS solution. The data provided from the MIDG includes orientation, position, velocity, acceleration, and angular rates. Figure 3.6 illustrates the MIDG hardware's input and output data. The sensor actually generates the INS data, which is combined with the GPS data when passed to the body worn computer.

When the MIDG is in IMU mode, the unit provides the most basic operation. In this mode, the MIDG II provides calibrated values for angular rate, acceleration, and magnetic field. Measurements from the GPS receiver are also available. However, none of the position/velocity/attitude estimation algorithms are executed. As a result, attitude is not available, and position and velocity are available directly from the GPS receiver at up to 5HZ.

The MIDG provides head tracking information to the body worn computer. This head tracking information updates the graphics which are displayed to the operator. If the operator is looking away from the LZ, an arrow indicator is displayed and directs the operator in the direction of the LZ and the tunnel in the sky display. The head tracker is mounted and oriented to the vertical position of the operator. The system is designed to operate when the operator has deployed the parachute and is in a vertical orientation. The MIDG weights under 55 grams, and provides orientation data via 3-axis rate gyros, 3-axis accelerometers and 3 axis magnetometer. In addition, position, velocity and attitude from a Kalman filter is available at a 50Hz rate. The MIDG can also operate in three distinct moods of operation: IMU, VG(vertical gyro), and INS. The IMU mode represents the most basic operation, none of the position, velocity or attitude estimation algorithms are executed. In VG mode, basic attitude estimations are added to the IMU mode. And finally in INS mode, the MIDG provides estimates of position, velocity and attitude up to 50Hz using a state estimation filter [MIC07].

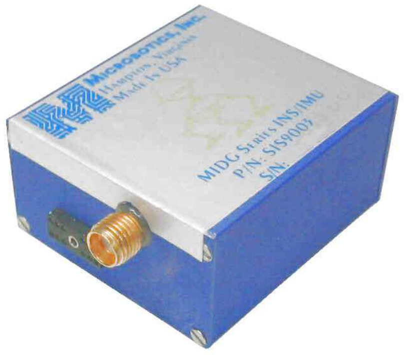


Figure 3.7: The MIDG II used for INS and head tracking functions.

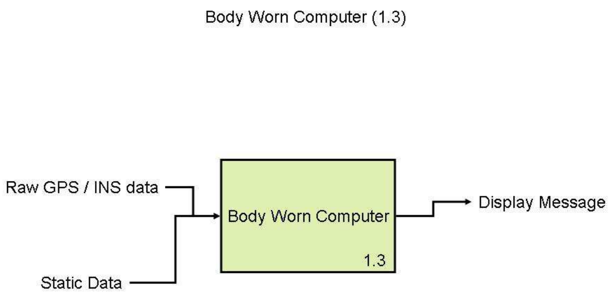


Figure 3.8: Overall high level GPS Navigational Display System.

3.3 Body Worn Computer

The body worn computer is the heart of the entire system and is composed of several components. These components include the main processing board, graphics chip, host connection board, and mounting equipment. The body worn computer provides all of the on board processing for the system. The MIDG tracking unit is connected to the body worn computer via a serial connection. The streaming data is decoded as described in Chapter 4 and is then used to generate the path solution. Once the navigation message is composed, the data generates the tunnel in the sky graphics. In addition to decoding the raw data, the CPU also stores the decoded

Figure 3.9: The Kontron GP8 is a general processing module which utilizes the Intel 80219 ARM architecture to deliver 600Mhz processing. In addition, the GP8 module also has an on-board SM-501 graphics chip to provide SVGA video output capability [KON06].

information in an onboard file for historical reference. After being decoded the data is stored to allow analysis of the solution utilized for the path generation.

3.3.1 Processing Board. The processing system must be small enough to fit into a paratrooper's ammo pouch. This allows the paratrooper to wear the computer on their body, while still being physically connected to the NVG input. In addition, the processing system must have the processing power to process the MIDG data and produce a valid navigational display. To achieve the minimal size for this proof of concept, and combine the head tracking and GPS signals into a visible navigational display, the Intel 80219 processor was selected as the on board processor.

The Intel 80219 is an ARM based processor, which is a 32-bit Reduced Instruction Set Computer (RISC) processor architecture developed by ARM Limited and is widely used in embedded systems. The ARM family of processors account for over 75% of all 32-bit RISC Central Processing Units (CPU), making it one of the most prolific 32-bit architectures in the world. ARM CPU's are found in consumer electronics from portable devices, such as Personal Digital Assistances (PDA), mobile phones, and calculators to computer peripherals such as hard drives and desktop routers [ARM05].

The 80219 processor runs at 600MHz and processes a 133 MHz Peripheral Component Interconnect (PCI)-X interface for a 1 gigabit per second (GB/s) throughput.

The internal bus operates at 200 MHz and 1.6 GB/s internal bandwidth. The 80219 General Purpose PCI Processor also features a 200 MHz Double Data Rate (DDR) Synchronous Dynamic Random Access Memory (SDRAM) controller with Error Correction Code (ECC) support up to 1 GB of 64-bit DDR SDRAM. It also supports 32-bit memory for applications that are more space sensitive. It contains a programmable, 31-bit local bus designed for embedded applications requiring connections to non-PCI peripheral components such as Application Specific Integrated Circuit (ASIC), Flash memory, or Digital Signal Processors (DSP). The Intel 80219 has additional features that accelerate I/O throughput. A 2-channel Direct Memory Access (DMA) controller facilitates increased PCI-to-memory and memory-to-memory throughput. The 80219 is compliant with ARM Version 5TE instruction set (excluding the floating point instruction set) [INT06]. The Intel 80219 is a 600Mhz Intel XScale ARM processor. The ARM processor is a reduced instruction set and has an onboard graphics chip. The distinct advantage of the onboard graphics chip is the ability to drive a color Super Video Graphics Array(SVGA) required by the AN/PVS-21 HUD. The 80219 processor chip is being utilized using the Kontron GP8 General Processing board, which provides the external memory and video graphics needed by this project.

3.3.2 Host Connection Board . The GP8 supports several input/output devices such as RS-232 serial connection, Universal Serial Bus (USB) host/client, and SVGA video output. Unfortunately the GP8 board is only a processing board and does not contain any onboard connectors to allow external devices to mount onto the system. To provide this external component connection service, the Diamond Point D2426 X-board baseboard is being utilized.

This board weighs .22 lbs and is 5.71" in length, 2.68" in width and .75" in overall height. The D2426 baseboard acts as the host "motherboard" for the GP8 by providing the external connections needed to accomplish the external interface [INT07b]. The D2426 baseboard was designed by Diamond Point International specifically to

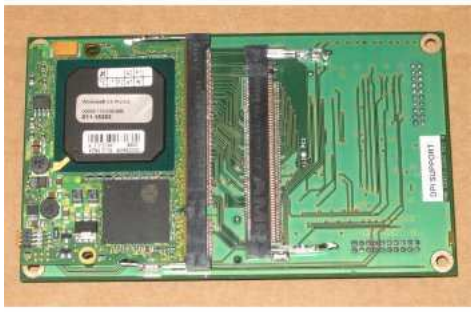


Figure 3.10: Diamond Point International D2426 X-board Baseboard utilized as the host board. The D2426 provides external connectivity with a small light-weight package. Photo shows D2426 with GP8 mounted on left side of the board [INT07b].

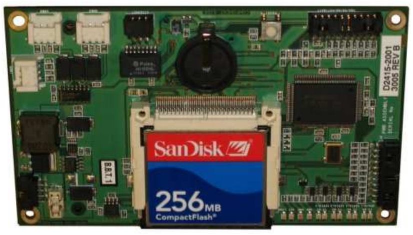


Figure 3.11: This photo shows the rear of the D2426 with an inserted CompactFlash card, plus the external mount points [INT07b].

operate with the Xboard family of general processing modules. The GP8 processing board mounts directly onto the D2426 board via an IDE connection. Figure 3.10 shows the overall size of the D2426 with a mounted GP8 module attached, and Figure 3.11 also shows the D2426, but from the rear of the board. Figure 3.11 also displays the size of the board with a compact flash card inserted. In addition to the compact flash card on the rear, the external mounting points, plus power connectors are visible.

3.4 Display

3.4.1 Graphics Card. The Kontron GP8 board comes with an on-board graphics chip (Silicon Motion SM501), which is necessary to provide the SVGA resolution required by the STS NVG e-module, to generate the HUD display. Figure

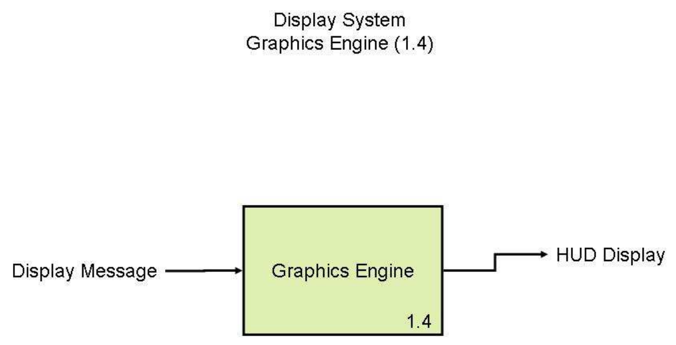


Figure 3.12: Display subsystem consisting of on-board graphics card and SVGA monitor (NVG HUD).

3.12, breaks out the display subsystem inputs and outputs. The graphics engine take in the display coordinates and generate the HUD display. The SM501 is a Mobile Multimedia Companion Chip (MMCC) device which connects to the PCI Bus. The SM501 removes the burden of 2D and video acceleration from the the Intel 80219 processor chip. By off-loading this additional workload, the GP8 module can handle the processing requirement to update the navigation display with up-to-date data.

3.4.2 Non-Computing Components.

3.4.2.1 GPS antenna. The system utilizes a small GPS antenna in order to receive the GPS signal. The antenna is mounted to the paratrooper's helmet to provide signal reception once the operator is in a vertical position. This is needed to ensure the maximum signal reception is achieved during the under canopy stage of the jump. The system was developed for use once the parachute deploys and the operator is in a vertical position.

Figure 3.13: Gentex PM jump helmet without any additional equipment such as NVG mounts or life support systems [GEN08].

3.4.2.2 Jump Helmet. The system can be mounted to any DoD approved helmet, but in this project, the system is designed to be mounted on the Gentex ParaMaster HALO/HAHO helmet (See Figure 3.13). This helmet is designed for the military operations dealing with HALO and HAHO jumps, which require the wearer to use life support systems at the jump altitudes. This is the latest approved jump helmet used by the US Army and is currently being utilized by its special operations and airborne personnel.

3.4.2.3 Power. The system is designed to operate under its own power and must therefore provide power to all associated components. The MIDG unit draws 1.2 watts, in addition to the GP8 board drawing 4 watts at 3.3 volts and the D2426 board drawing an additional 5 volts. Because of this high power consumption, light weight requirement, and long life operation, the 12 volt Lithium Polymer battery was chosen in a dual stack configuration. This provides enough power to operate the entire system for nearly 2 hours.

3.4.3 HUD Display. Sensor Technology Systems utilizes Tek Gear's eMagin-based O2 microdisplay OEM kit for its Enhanced Heads-Up Display (E-HUD) module on the AN/PVS-21 Low Profile Night Vision Goggle. The O2 OEM Kit is a drop in module that is designed to be integrate into high resolution microdisplays. The O2 Kit includes eMagins SVGA+ Organic Light Emitting Diode (OLED)-XL microdisplays and high performance controllers.

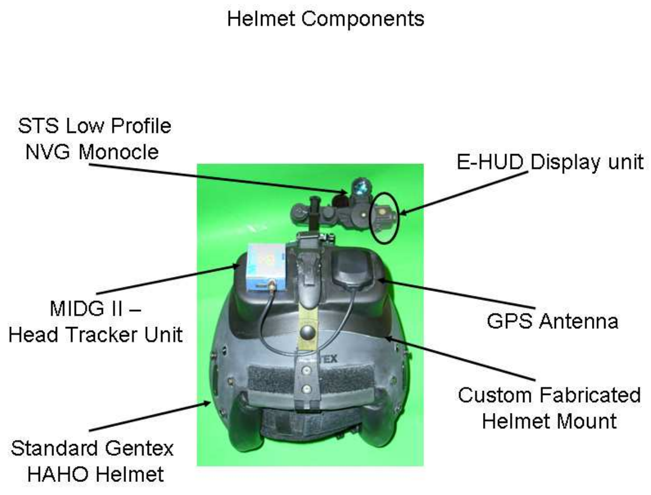

Helmet Components
STS Low Profile NVG Monocle
E-HUD Display unit
MIDG II – Head Tracker Unit
GPS Antenna
Custom Fabricated Helmet Mount
Standard Gentex HAHO Helmet

Figure 3.14: Helmet Components.

To provide the operator with the appropriate navigation display information, the system utilizes a low-level graphics library to draw each visual graphics component, then this information is passed to the HUD display system. The STS NVG HUD system operates on an SVGA input, therefore, the critical portion of the graphic output process is to ensure the graphic components, such as the tunnel in the sky, are correctly located and generated within the given display area in the SVGA format.

3.5 Completed System

Figures 3.14 through 3.17 are photos of the completed assembled system. Figure 3.14 is the head worn portion of the system. This is composed of the standard Gentex HAHO jump helmet, with a custom fabricated head tracker and GPS antenna mount. The fabrication of the helmet mount allowed for the sensors to have a stable and level mounting point on the helmet. Also shown in the photo is the attached STS low profile monocle with the attached E-HUD display unit. The E-HUD system is the component which generates the visual display for the operator.

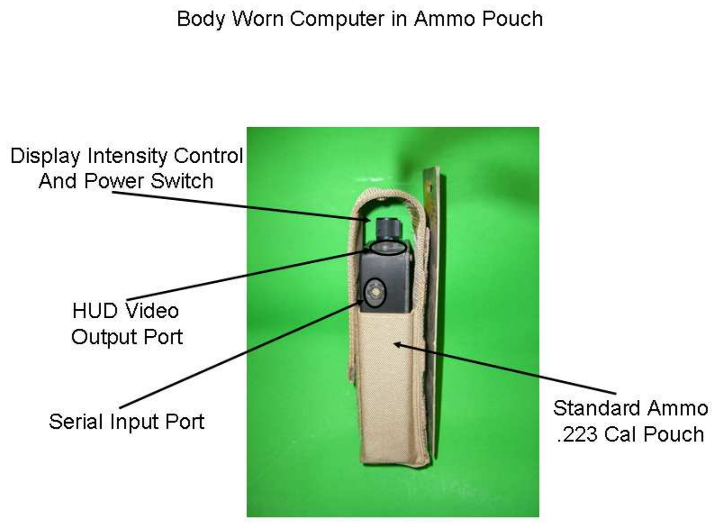

Body Worn Computer in Ammo Pouch
Display Intensity Control And Power Switch
HUD Video Output Port
Serial Input Port
Standard Ammo .223 Cal Pouch

Figure 3.15: Body Worn Computer in a .223 Cal Ammo Pouch.

Figures 3.16 and 3.15 show the body worn computer. This portion of the system is designed to fit in a standard .223 caliber ammo pouch as is shown in figure 3.15, while still providing access to controls and allowing I/O cables to exit without modification to the pouch. Figure 3.16 illustrates the dimensions of the total enclosure, plus highlights the external component. For example, across the top is a display intensity control knob, the power switch, and the HUD video out port. Down the right side is the serial input port and the compact flash port.

Figure 3.17 shows the various internal products which make up the body worn computer. The 12 volt batter has been removed to allow visual access to the various internal circuit boards. The entire system is mounted on a Diamond D2426 base board for power and I/O connectivity. The TekGear E-HUD circuit board is mounted towards the top side of the container to provide the E-HUD video display output. And finally, the GP8 processor module is on the bottom half of the container.

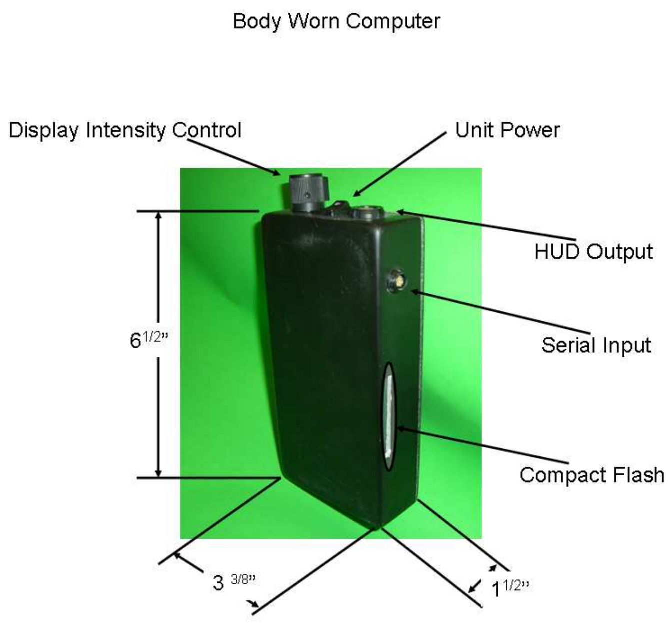


Figure 3.16: Body Worn Computer Diagram.

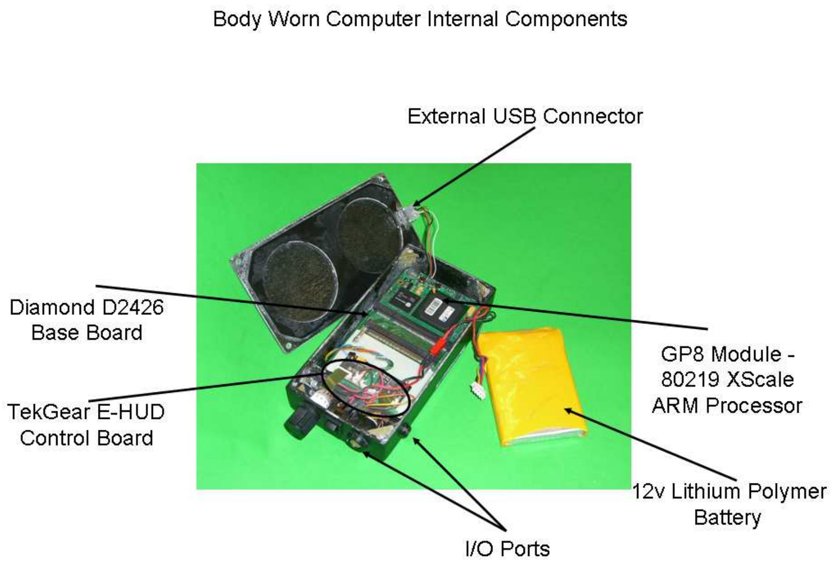

Body Worn Computer Internal Components
External USB Connector
Diamond D2426 Base Board
TekGear E-HUD Control Board
GP8 Module - 80219 XScale ARM Processor
12v Lithium Polymer Battery
I/O Ports

Figure 3.17: View of internal Body Worn Component.

3.6 Summary

Since the first time humans began utilizing parachutes, research has been conducted on how to make them more efficient, travel further, and carry more weight. With the advent of HAHO jumps, paratroopers can cover dozens of miles during an single jump. This is of great value to the military because it allows for insertions of airborne personnel into areas without endangering aircraft. With the added concealment of night, these jumps can become even safer. The equipment describe in this chapter provides and overview of the components which are needed to accomplish the task of adding a navigational aid to night vision goggles allowing operators to travel vast distances safely navigating to their final destination. There are several alternative technologies which are currently being explored, but nothing currently being developed provides the necessary information in a small compact easy to use package. This research attempts to bridge this gap be developing a system which gives the operator clear navigational information while not adding excessive weight.

IV. Software Implementation

This chapter describes the software and the engineering methods used to implement the operational software utilized by this project. The software is designed to take an input data stream from a sensor (MIDG) and calculate an accurate navigational path for the paratrooper, based on environmental factors, such as wind, parachute type and travel distances.

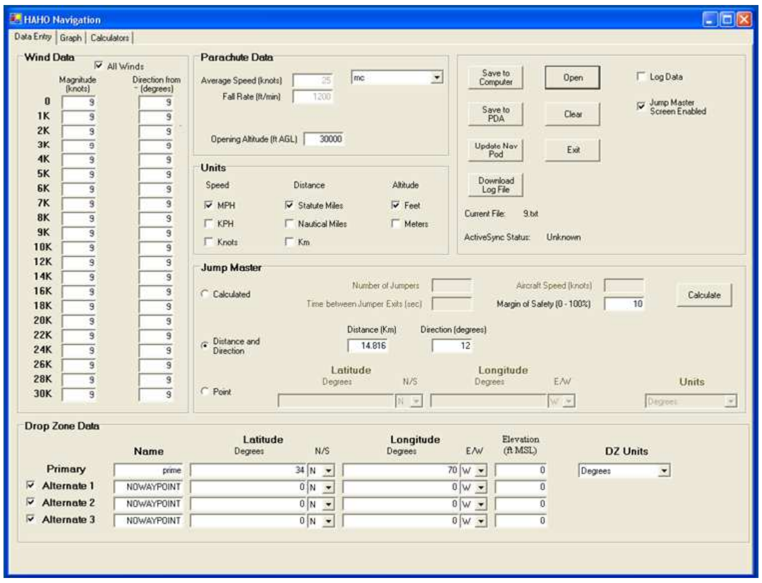
4.1 Input Data

The initial phase of the program begins by initializing the MIDG 2 sensor to set unit parameters, such as message frequency and serial settings. Once set, the MIDG begins streaming data via a serial connection into the body worn computer. This program then parses the message data being received. The program stores the latitude, longitude, altitude, yaw, pitch, roll, and magnetic orientation data in an output file with the "raw" extension. This data can be utilized to provide record and debugging information for future development. All latitude, longitude, and altitude data is then transformed into an East/North/Up (ENU) coordinate system.

The ENU coordinate system is a local cartesian coordinate system which sets the origin point (known as [0,0,0]) to the landing zone. This enables all calculations to be performed in a local Cartesian coordinate system.

4.2 Jump Parameters

The program's next step is to start a second thread to run the calculation functions. The system is designed to run two user level threads simultaneously. The first thread reads the sensor input and parses the data into usable messages (as described above). This thread also stores the data into a log file and into a data repository structure, utilized by the second thread. The critical sections (writes / reads) are protected by a set of locks which only allows one thread access at a time. Once the data is written and the locks are released by the sensor thread, the calculation thread can then lock the data, to perform its read.


HAHO Navigation

Data Entry | Graph | Calculators

Wind Data [✓ All Winds]

	Magnitude (knots)	Direction from (degrees)
0	9	9
1K	9	9
2K	9	9
3K	9	9
4K	9	9
5K	9	9
6K	9	9
7K	9	9
8K	9	9
9K	9	9
10K	9	9
12K	9	9
14K	9	9
16K	9	9
18K	9	9
20K	9	9
22K	9	9
24K	9	9
26K	9	9
28K	9	9
30K	9	9

Parachute Data

Average Speed (knots): 25 — mc
Fall Rate (ft/min): 1200
Opening Altitude (ft AGL): 30000

Units

Speed: [✓ MPH] KPH Knots
Distance: [✓ Statute Miles] Nautical Miles Km
Altitude: [✓ Feet] Meters

Current File: 9.txt
ActiveSync Status: Unknown

[ ] Log Data
[✓ Jump Master Screen Enabled]

Jump Master

( ) Calculated — Number of Jumpers: — Time between Jumper Exits (sec): — Aircraft Speed (knots): — Margin of Safety (0 - 100%): 10

(● Distance and Direction) — Distance (Km): 14.816 — Direction (degrees): 12

( ) Point — Latitude Degrees, N/S: N — Longitude Degrees, E/W: W — Units: Degrees

Drop Zone Data

	Name	Latitude Degrees	N/S	Longitude Degrees	E/W	Elevation (ft MSL)
Primary	prime	34	N	70	W	0
✓ Alternate 1	NOWAYPOINT	0	N	0	W	0
✓ Alternate 2	NOWAYPOINT	0	N	0	W	0
✓ Alternate 3	NOWAYPOINT	0	N	0	W	0

DZ Units: Degrees

Figure 4.1: HAHO Mission Planner main screen

The first step of the calculation thread is to read the jump parameters. This consists of two files. The first is a text file which overrides default parameters, such as parachute data, final approach characteristics, and landing zone variables. An example parameter file is located at Appendix A. This data allows each jump to be configured according to the specific circumstances and operator's preferences.

The second file loaded into the system is the wind data file. This file is a standard file generated by the two major air delivery programs currently utilized. The first program is called the Joint Precision Airdrop System Mission Planner (JPADS) [DRA06]. This system is utilized by both the Air Force and the Army to deliver both personnel and cargo to specific landing zones. The system receives wind data via a radio signal from an air drop sensor, which is then fed to an on-board laptop to generate the wind data file. The second program used by the air delivery community is the HAHO Mission Planner, which is built specifically for the HAHO missions. Figure 4.1 is a screen shot of the HAHO Mission Planner's main input panel.

4.3 Algorithms

The navigation software consists of three major algorithms: the wind calculations, the calcSituation, and the display algorithm. The wind calculation algorithm processes the input wind data and the parachute parameters to produce an accurate navigation solution. The calcSituation algorithm utilizes the current positional information to produce an updated situational report. This algorithm is where the delta positional information is calculated and passed on to the display algorithm. The final algorithm developed is the display algorithm, which transforms the physical world into the NVG lens vector to create a screen display.

4.3.1 Wind Calculations. The wind algorithm reads wind data (wind speed and direction) for every level loaded on the wind 1D file. Once the entire wind file is loaded and set, the system generates three different paths. The first path is from the Landing Zone to the Final Approach Point (FAP). The next path is generated from the FAP to the Aim Point (AP). The final path is the path from the initialization point to the AP. Each of these points is calculated based on inputs from the wind file and the configuration text file. This solution guides the paratrooper on an offset to compensate for wind drift, and allows the operator to reach the desired waypoint in space.

Figure 4.2 illustrates the decision flow process followed in determining the positional data. Figure 4.3 is a graphical display of the effects winds have upon the jumper. In Figure 4.3, the jumper heads to the aim point. The effects of the wind will cause the jumper to drift to the desired physical way point. The surrounding circles represent the three ranges of the parachute travel ability. The solid middle circle is the ideal glide slope, where the inner dashed circle represents the possibility of the parachutist overshooting the intended target. The outer dashed circle represents the maximum travel distance for that particular parachute. If the jumper is beyond the outer circle, the target is out of reach, even at maximum glide.

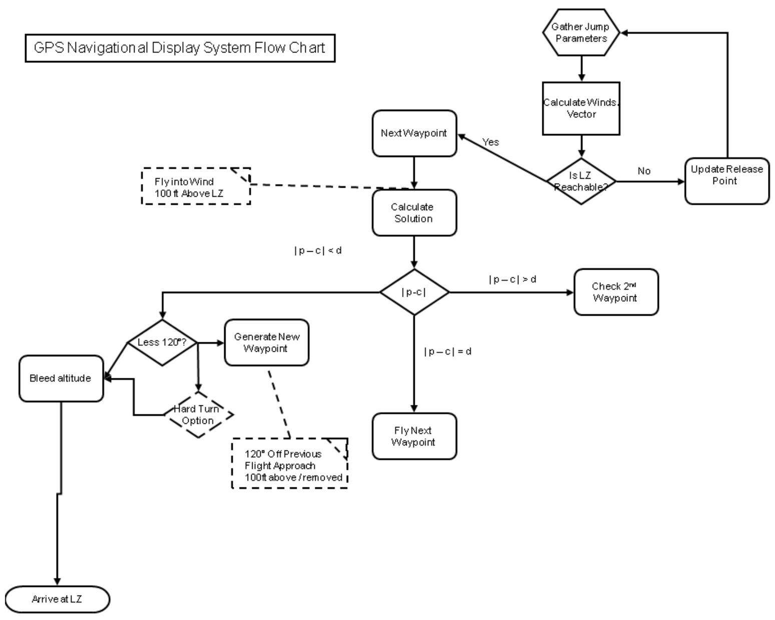


Figure 4.2: System flow chart.

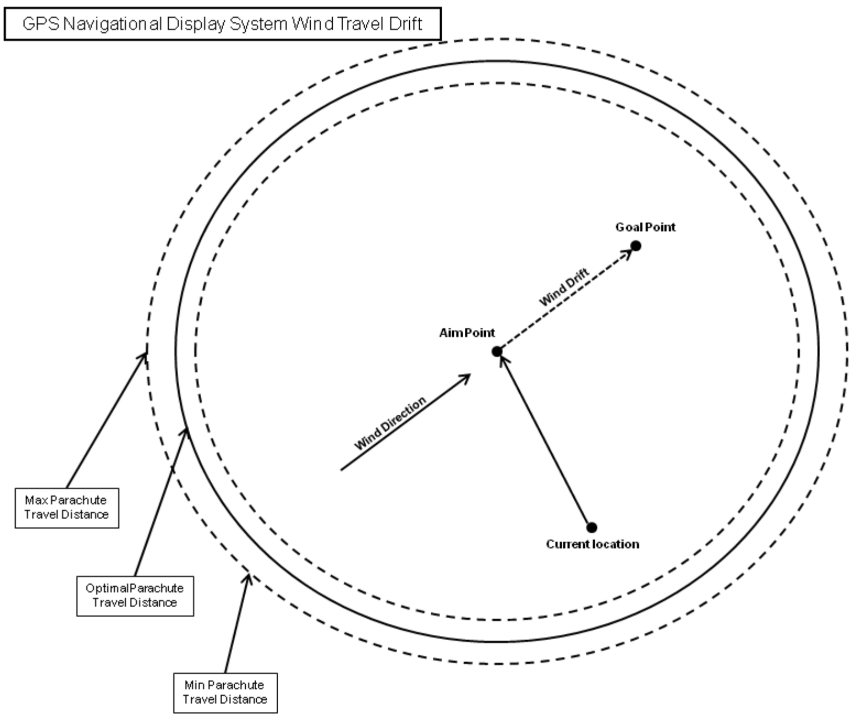

GPS Navigational Display System Wind Travel Drift
Goal Point
Wind Drift
Aim Point
Wind Direction
Current location
Max Parachute Travel Distance
Optimal Parachute Travel Distance
Min Parachute Travel Distance

Figure 4.3: Wind drift effects on jumper. Jumper's heading directs them to aim point, compensating for wind effects.

Because the speed and direction of each wind level is different, the amount of time the jumper spends in each level determines wind drift effects. For example, if the winds at level two are blowing at 10 mph, out of the north, and the jumper spends 10 minutes within the wind level, then the jumper will have travelled 1.6 miles to the south, while in level two (See Equation 4.1).

$$10(minutes) * .16(miles/perminute) = 1.6(miles) \tag{4.1}$$

Equation 4.2 is the distance equation used to determine the distance, where d is distance, v is the velocity of the jumper and t is the amount of time spent within that wind level.

$$vt = d \tag{4.2}$$

$$\begin{aligned} v &= \text{velocity of jumper} \\ t &= \text{amount of time spent in wind level} \\ d &= \text{distance covered} \end{aligned}$$

Equation (4.2) shows how the calculations used to determine the proper wind drift to be added to the overall wind.

Equation 4.3 represents the calculation used to determine the amount of wind drift within a single level.

$$W = V_w \Delta t \tag{4.3}$$

$$
\begin{aligned}
W &= \text{drift due to wind} \\
V_w &= \text{2 dimensional wind velocity vector} \\
\Delta t &= \text{time in the wind level}
\end{aligned}
$$

In Equation 4.3 W is the drift due to wind, V_w is the 2 dimensional wind velocity vector, and Δt is the time in the wind level. Δt is determined by Equation 4.6.

For several layers, Equation 4.4 is used to calculate the total amount of wind drift.

$$
\underline{W}_{TOT} = \sum_{j=1}^{N} (\underline{V}_{wj} \Delta t_j) \tag{4.4}
$$

$$
\begin{aligned}
\Delta t_j &= \text{time within the } j^{\text{th}} \text{ level} \\
\underline{V}_{wj} &= \text{is the wind velocity vector in the } j^{\text{th}} \text{ level}
\end{aligned}
$$

Wind level j is defined by h_{min} and h_{max}, the minimum and maximum height in that level. We want to calculate wind effect over altitude interval $h_{max} - h_{min}$. Figure 4.4 is a graphical example of the level description.

For a parachute in descent, Equation 4.5 is used to determine the downward movement distance.

$$
\Delta h = V_d \Delta t \tag{4.5}
$$

$$
\Delta t = \frac{\Delta h}{V_d} \tag{4.6}
$$

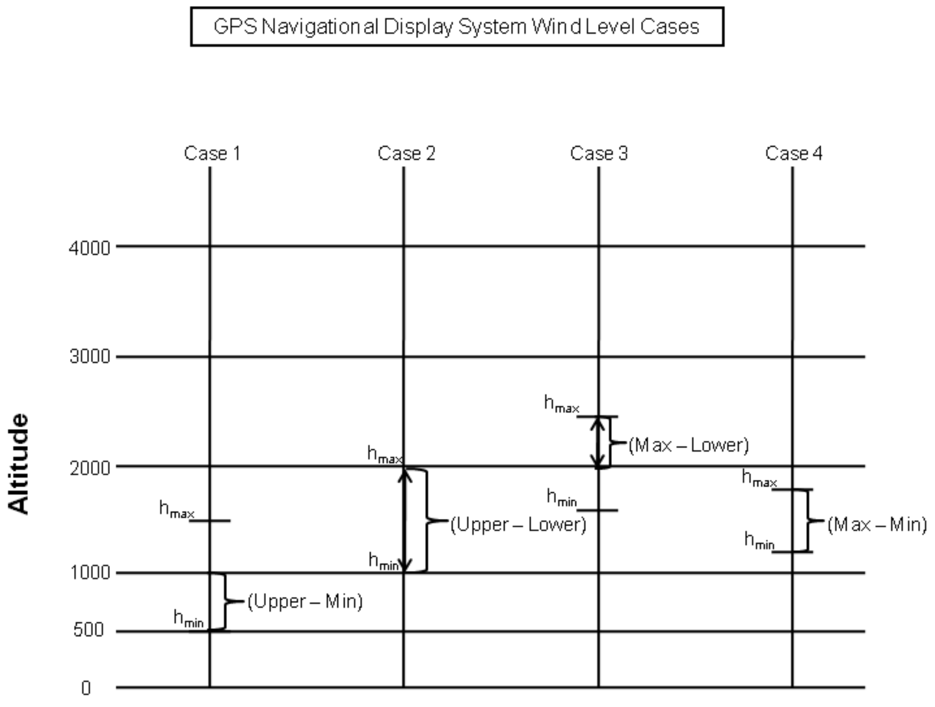


Figure 4.4: Display of four wind cases used to calculate navigational path. Case 1: Minimum height is below current wind level. Case 2: Minimum height and maximum height are above and below the wind level boundaries. Case 3: Maximum height is above current wind level. Case 4: Minimum and maximum heights are within same wind level.

$$\begin{aligned}\Delta h &= \text{downward movement distance}\\ V_d &= \text{descent velocity}\\ \Delta t &= \text{time of descent}\end{aligned}$$

There are four cases which are used to determine the time delta utilized in the calculation. These four cases are dependant on the maximum and minimum height of the situation. For a given wind layer, Equation 4.7 describes the determination of Δh, for a given wind level. This equation is also described visually in Figure 4.4.

$$\Delta h_j = \begin{cases} h_{maxj} - h_{min} \text{ if } h_{max} > h_{maxj} \text{ and } h_{min} \geq h_{minj} \\ h_{maxj} - h_{minj} \text{ if } h_{max} \geq h_{maxj} \text{ and } h_{min} \leq h_{minj} \\ h_{max} - h_{minj} \text{ if } h_{max} \leq h_{maxj} \text{ and } h_{min} \leq h_{minj} \\ h_{max} - h_{min} \text{ if } h_{max} < h_{maxj} \text{ and } h_{min} > h_{min} \end{cases} \tag{4.7}$$

$$\begin{aligned}\Delta h_j &= \text{downward movement distance}\\ h_{max} &= \text{maximum altitude}\\ h_{min} &= \text{minimum altitude}\\ h_{maxj} &= \text{upper boundary of wind level j}\\ h_{minj} &= \text{lower boundary of wind level j}\end{aligned}$$

Equation 4.8 is used to determine the Δt_j for each level, where Δh_j is the downward movement distance, and V_d is the descent velocity.

$$\Delta t_j = \frac{\Delta h_j}{V_d} \tag{4.8}$$

$$\begin{aligned}\Delta t_j &= \text{time within the level}\\ \Delta h_j &= \text{downward movement distance}\\ V_d &= \text{descent velocity}\end{aligned}$$

Equation 4.8 now provides the data to calculate the $\underline{W}_{TOT}$, as described in Equation 4.4.

4.3.2 CalcSituation Algorithm. The calcSituation algorithm is the portion of the system which provides the updated positional information and determines the positional delta. Figure 4.5 is the visual representations of Equations 4.9 through 4.15 used to produce the delta position in relation to the projected path.

Within Equation 4.9, $\underline{b}$ is the projected distance, while $\underline{a}$ is the true distance. $\underline{P}_{ref}$ is the established fixed location, such as the final approach point or initial release point. This is the same fixed point which will be used throughout the entire cycle of calculations until the next phase of the descent is entered. The final variable, $\underline{C}$ is the projected aim point taking into consideration the effects of wind drift.

$$\underline{b} = \underline{a}\frac{\underline{C} - \underline{P}_{ref}}{\|\underline{C} - \underline{P}_{ref}\|} \tag{4.9}$$

$$\begin{aligned}\underline{b} &= \text{projected vector } \underline{p} \text{ and } \underline{P}_{ref}\text{projected onto desired true path}\\ \underline{a} &= \text{vector between } \underline{p} \text{ and } \underline{P}_{ref}\\ \underline{P}_{ref} &= \text{established fixed location}\\ \underline{C} &= \text{true aim point}\end{aligned}$$

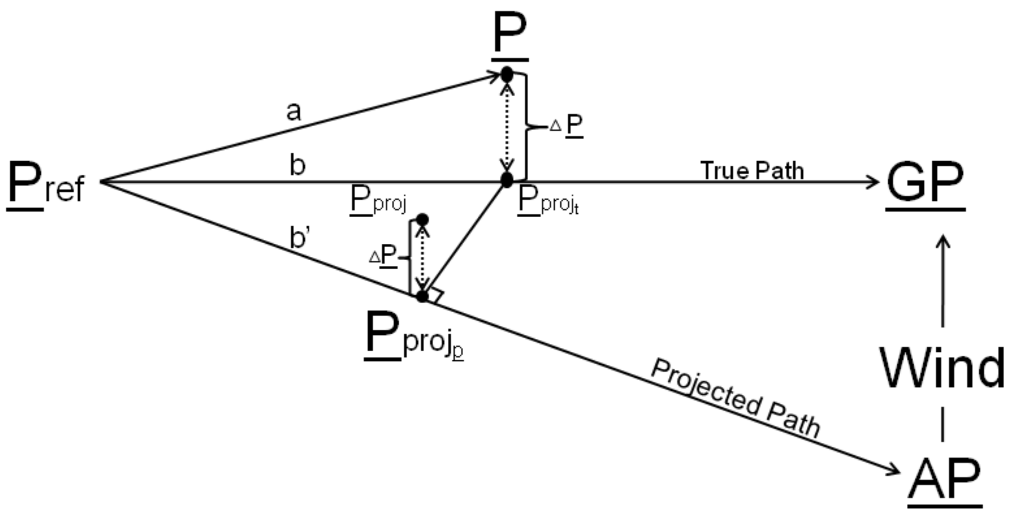


Figure 4.5: This figure shows the visual representation of the components used to establish the positional delta from the true path.

Equation 4.10 is used to establish a, which is the true distance along the true path. $\underline{p}$ represents the true position, with $\underline{P}_{ref}$ being the same fixed point used in Equation 4.9. See Figure 4.5 for a visual description of the variables used for this calculation.

$$\underline{a} = \underline{p} - \underline{P}_{ref} \tag{4.10}$$

$$\begin{aligned} \underline{a} &= \text{true distance} \\ \underline{p} &= \text{true position} \end{aligned}$$

Equation 4.11 defines the projected location at time t, where $\underline{b}$ is the value established in Equation 4.9 and $\underline{P}_{ref}$ is again the same fixed location used in previous calculations.

$$\underline{P}_{proj_t} = \underline{P}_{ref} + \underline{b} \tag{4.11}$$

$$\underline{P}_{proj_t} = \text{projected position at time } t$$

$\Delta\underline{p}$ is established in Equation 4.12 as the difference between the product of the present point ($\underline{p}$) subtracted from the projected point ($\underline{P}_{proj_t}$) established in Equation 4.11.

$$\Delta\underline{p} = \underline{p}_{ref} - \underline{P}_{proj_t} \tag{4.12}$$

$\Delta \underline{p}$ = delta position projected on the projected path

For Equations 4.13 through 4.15 refer to Figure 4.5. Equation 4.13 establishes $\underline{b}'$, as the distance on the projected path to location $\underline{c}$ (projected aim point). The equation multiplies the difference between $\underline{c}$ and $\underline{P}_{ref}$ (which is the same reference location used previously), by the ratio of the magnitude of $\underline{b}$, which was established in Equation 4.9, and the magnitude of the aim point ($\underline{AP}$), and the established reference point($\underline{P}_{ref}$).

$$\underline{b}' = (\underline{c} - \underline{P}_{ref})\frac{\|\underline{b}\|}{\|\underline{AP} - \underline{P}_{ref}\|} \tag{4.13}$$

$\underline{b}'$ = distance on the projected path to location $\underline{c}$

$\|\underline{b}\|$ = magnitude of projected location established in Equation 4.9

$\underline{AP}$ = projected aim point, considering wind drift

Next in the Equation $\underline{P}_{proj_p}$ is established as the sum of $\underline{P}_{ref}$ (same as previously used) plus $\underline{b}'$(established in Equation 4.13).

$$\underline{P}_{proj_p} = \underline{P}_{ref} + \underline{b}' \tag{4.14}$$

$\underline{P}_{proj_p}$ = projected position in relation to projected path

The final equation in the calcSituation establishes the projected position ($\underline{P}_{proj_p}$) from Equation 4.14 plus the delta position ($\Delta \underline{p}$) established in Equation 4.12.

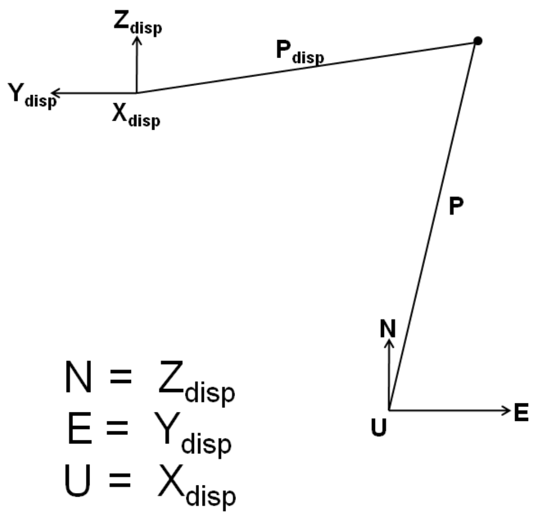


Figure 4.6: Rotate

$$\underline{P}_p = \underline{P}_{proj_p} + \Delta\underline{p} \tag{4.15}$$

$\underline{P}_p$ = current position with the goal at $\underline{\text{C}}$ (or aim point)

4.3.3 Display Algorithm. The third algorithm which is used by the system, to generate the navigational message for the operator, is the display algorithm. This algorithm is responsible for transforming a 3-D coordinate into the pixel location. This physical location is defined by the transformed operator's NVG position (lens position) and the target location. Figure 4.6 describes the transformation utilized by the system in which the lens position's (P) x,y, and z axes are related to the physical world's x,y, and z axes.

The axis transformation is performed by applying Equations 4.16 through 4.20. This set of equations is applied to every position which must be mapped to the screen.

Equation 4.16 calculates the vector below point location ($\underline{X}^{ENU}$), which needs to be projected and the jumper's projected position ($\underline{P}^{ENU}_{proj_p}$), established in Equation 4.15.

$$\Delta\underline{X}^{ENU} = \underline{X}^{ENU} - \underline{P}^{ENU}_{proj_p} \tag{4.16}$$

$\Delta\underline{X}^{ENU}$	=	delta from projected jumper position to projected positional point
$\underline{X}^{ENU}$	=	point to project
$\underline{P}^{ENU}_{proj_p}$	=	position from Equation 4.15 transformed into ENU coordinates

The next step is to rotate the $\Delta\underline{X}^{ENU}$ into display (I) frame, by multiplying the $\Delta\underline{X}^{ENU}$ by the direction cosine matrix.

$$\Delta\underline{X}^{I} = \underline{C}^{I}_{ENU}\Delta\underline{X}^{ENU} \tag{4.17}$$

$\Delta\underline{X}^{I}$	=	position x, y, z rotated into image frame
$\underline{C}^{I}_{ENU}$	=	direction cosine matrix
$\Delta\underline{X}^{ENU}$	=	delta position from Equation 4.16

Equation 4.18 is the $\underline{C}^{I}_{ENU}$ matrix where, ψ represents yaw, θ represents pitch, and ϕ represents roll, which are obtained from the MIDG.

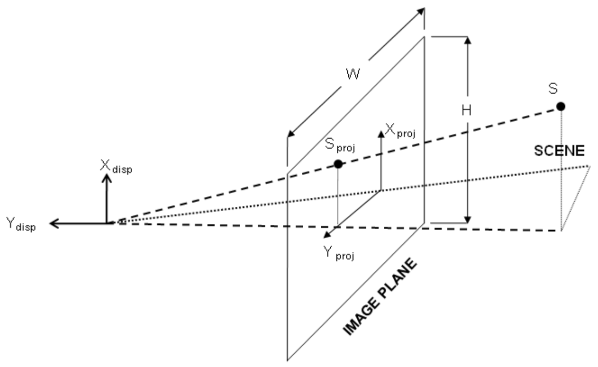


Figure 4.7: The image plane which rotates the coordinate from the physical world to the image plane.

$$\underline{C}^{I}_{ENU} = \begin{bmatrix} \sin\phi\cos\psi - \cos\phi\sin\theta\sin\psi & -\sin\phi\sin\psi - \cos\phi\sin\theta\cos\psi & \cos\phi\cos\theta \\ \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & -\sin\phi\cos\theta \\ \cos\theta\sin\psi & \cos\theta\cos\psi & \sin\theta \end{bmatrix} \tag{4.18}$$

Once the vector has been rotated into the display frame, the next step is to project the location into normalized pixel locations. Figure 4.7 shows the layout of the display in relation to Equations 4.19 and 4.20. The screen normalization takes the three dimensional rotated ENU position, and produces an $X\underline{pix}$ and $Y\underline{pix}$ location, based on Equations 4.19 and 4.20. These $X\underline{pix}$ and $Y\underline{pix}$ positions represent the final display screen location of the pixel to be projected.

$$X_{pix} = C_x \frac{X(1)}{X(3)} + \frac{M+1}{2} \tag{4.19}$$

X_{pix}	=	screen X pixel value
C_x	=	normalized screen ratio, based on display resolution
$X(1)$	=	X projected value of $\underline{X}^{Proj}$ from Equation 4.17
$X(3)$	=	Z projected value of $\underline{X}^{Proj}$ from Equation 4.17
M	=	number of horizontal pixels

$$Y_{pix} = C_y \frac{X(2)}{X(3)} + \frac{N+1}{2} \tag{4.20}$$

Y_{pix}	=	screen Y pixel value
C_y	=	normalized screen ratio, based on display resolution
$X(2)$	=	Y projected value of $\underline{X}^{Proj}$ from Equation 4.17
$X(3)$	=	Z projected value of $\underline{X}^{Proj}$ from Equation 4.17
N	=	number of vertical pixels

4.4 Software Components

The following section describes the software components that make up the Navigational application software. The first components described make up the wind calculation algorithm. As described in section 4.3.1, this performs the wind adjustments for the navigational message. The second set of software components make up the calcSituation algorithm (See Paragraph 4.3.2). The final software components implement the display algorithm as described in Paragraph 4.3.3. These three software

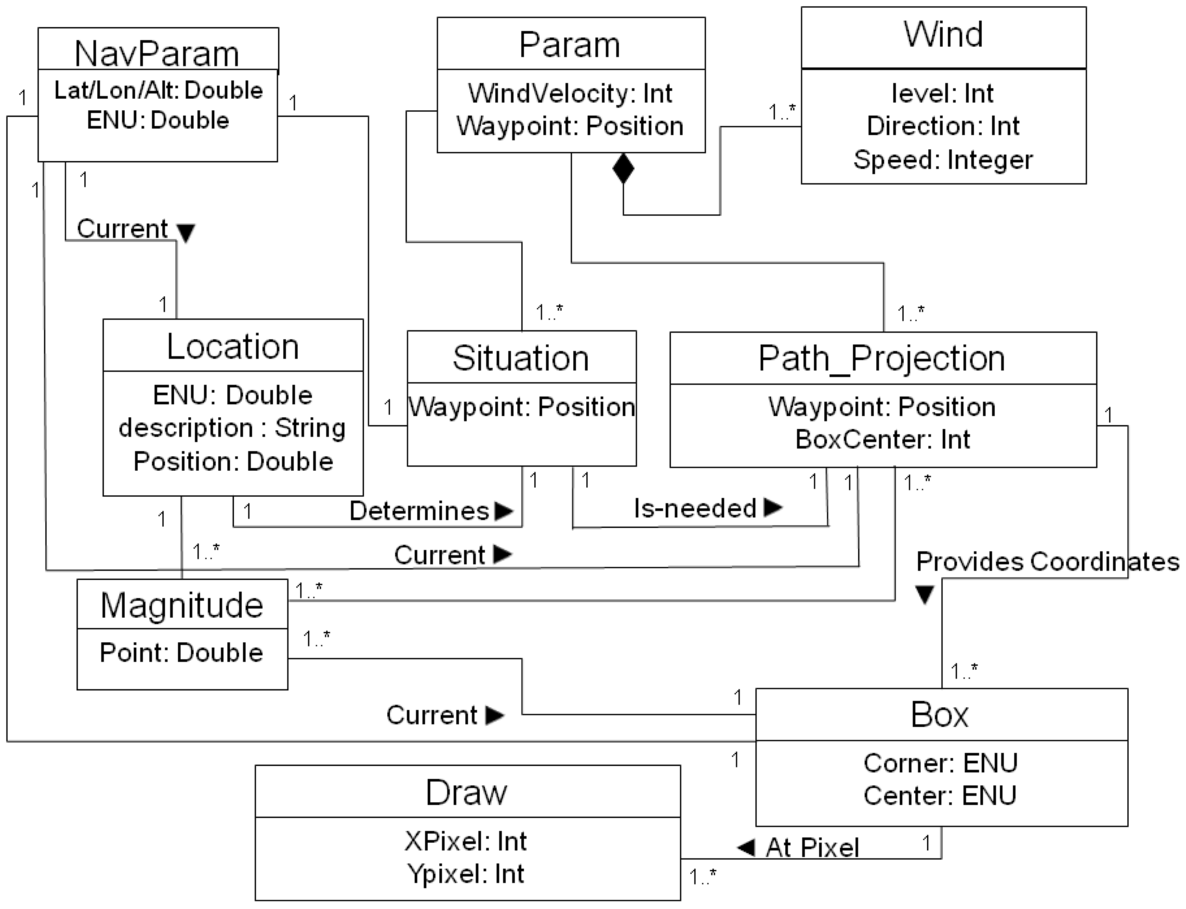


Figure 4.8: System domain diagram which describes the interaction between the software components.

sections compose the model/view/controller architecture for the application software. Figure 4.8 is the system domain diagram which describes the interaction between the software components.

4.4.1 Wind Algorithm Software. The wind algorithm is composed of the following two software components which provide the system with the wind calculation tables for use in generating a navigational solution which can be successfully reached.

4.4.1.1 Wind.cpp. The wind.cpp handles all the wind calculations that are used throughout the navigation path generation. Wind.cpp creates wind tables for every altitude level which is read in by the paramloader.cpp. The levels are stored as vectors which fluctuate to meet the needs for the number of wind levels needed for a particular mission. Once the tables are created, the information is used by the software to generate the navigational path between a reference point (i.e. the Aim Point) and the target point (i.e. Final Approach Point), by the calcSituation algorithm (See Paragraph 4.3.2). The winds for the altitudes between these two points are used to determine the amount of drift which will be encountered during the time within that particular wind level. This algorithm is described in much more detail in section 4.3.1. Figure 4.9 is the class diagram for the wind class. This class is composed of two main structures, params, and windVelocities. In addition, the wind header file acts as an interface used to access the wind methods.

4.4.1.2 paramloader.cpp. The paramloader.cpp performs the reads from both the config.txt and navaid.txt files to load system parameters and 1D wind file. Figure 4.10 is the class diagram for the paramloader. This class is composed of three structures, params, NavParams, and enuData; and three methods used to perform the various loads. The first method reads data from the config.txt file, which parses the data based on text file rules and keywords. If the inputs are invalid, an error is posted citing the error and the system halts and exits. The second method reads in the navaid.txt file, which is the 1D wind file standard used throughout the air delivery community. This method parses the data, based on comma separated values, and performs any data transformation needed on the input values, such as converting the latitude and longitude from degrees to radians and converts the directional inputs from compass headings to mathematical coordinates. These conversions are needed to perform all future equations utilizing this data. All values which are read in by both these classes are loaded into the Params data structure, (See Paragraph 4.5.0.4), to be used as the model portion of the system architecture. The paramloader is executed only once, at system initialization, since this data is static and therefore is never changed throughout the system operation.

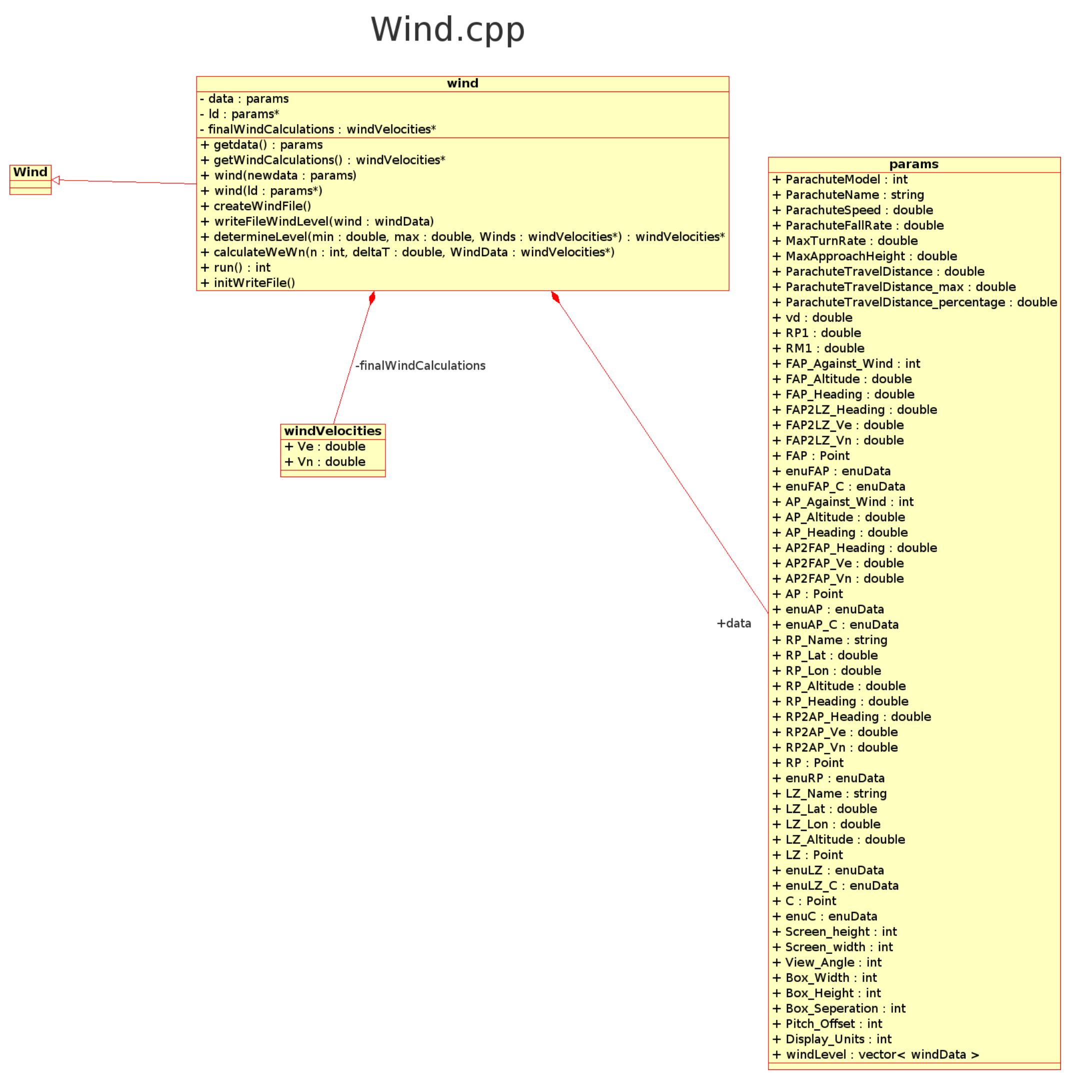


Figure 4.9: The wind class handles all the wind calculations that are used throughout the navigation path generation. This class is composed of two main structures, params and windVelocities. Additionally, the wind header file acts as an interface used to access the wind methods.

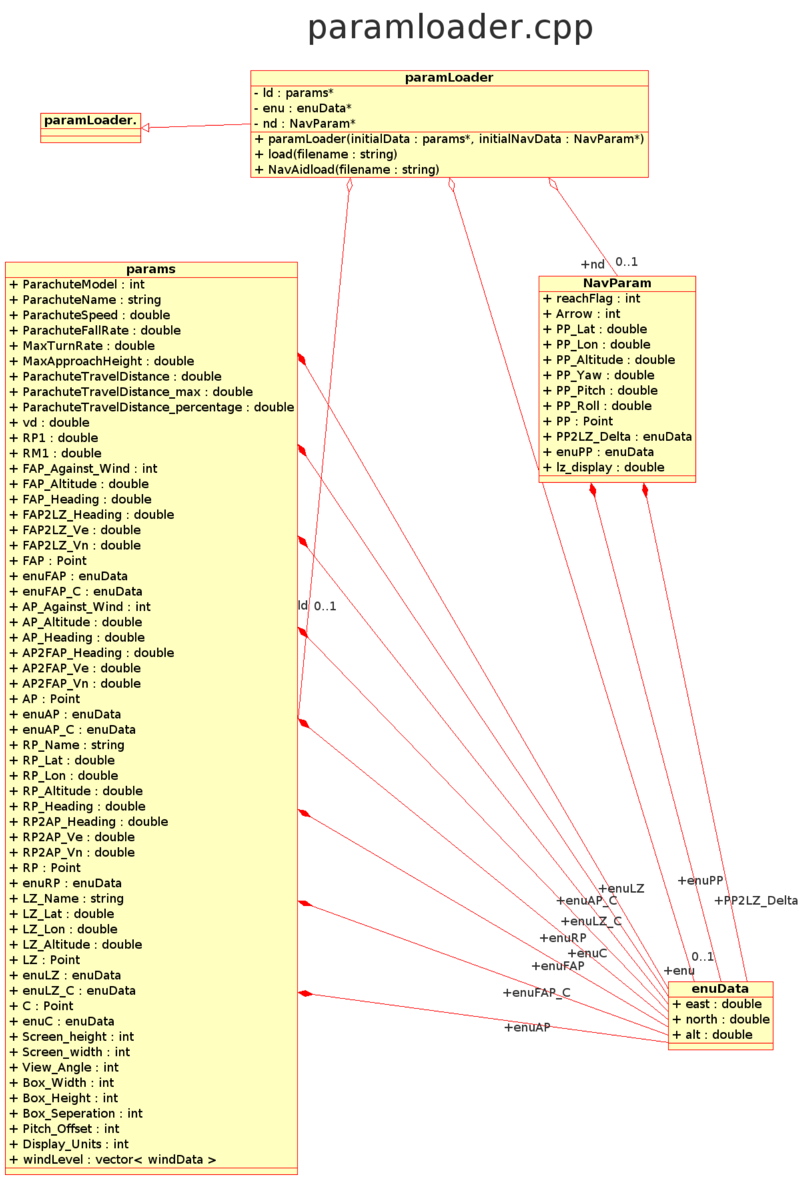


Figure 4.10: The paramloader.cpp performs the reads from both the config.txt and navaid.txt files to load system parameters and 1D wind file. This class is composed of three structures; params, NavParams, and enuData; and three methods used to perform the various loads.

4.4.2 CalcSituation Algorithm Software. These software components make up the CalcSituation (See Paragraph 4.3.2) algorithm portion of the system. These components parse incoming data reports and provide updated positional information to the display algorithm.

4.4.2.1 MIDGIIController.cpp. The MIDGIIController.cpp was first developed by Kresge [FLE07] utilizing the MIDG 2 sensor. The original code was modified to meet the needs of this software application. This class initializes the MIDG, the serial port, and the output record files. The MIDGIIController then begins to read incoming messages on the serial port and stores the data in the appropriate data structure for use by the processing segment of the software. The serial processing is performed in a separate thread which locks and unlocks the data structures as needed to ensure race conditions are avoided. This class provides the updated positional information which is used by the calcSituation (See Paragraph 4.3.2), by updating the NavParams (See Paragraph 4.5.0.5) with the positional and orientation data. This class also controls the rate at which data is read from the MIDG sensor, which is currently set at 2 Hz.

4.4.2.2 LLAConvert.cpp. LLAConvert.cpp is a utility class which performs the math functions to convert latitude and longitude into the east/north/up coordinates. East/north/up (ENU) Cartesian coordinates use the landing zone as the [0,0,0], to base all other location calculations. This allows for the math functions to be performed in a simpler manner, as compared to performing the same operations on latitude and longitude coordinates, which would be much more complex.

4.4.2.3 location.cpp. To generate the proper display, the present position information is read from a data structure NavParam (See Paragraph 4.5.0.5) containing the latitude, longitude, altitude, yaw, pitch and roll for the operator's current position and orientation. Figure 4.12 is the class diagram for Location.cpp,

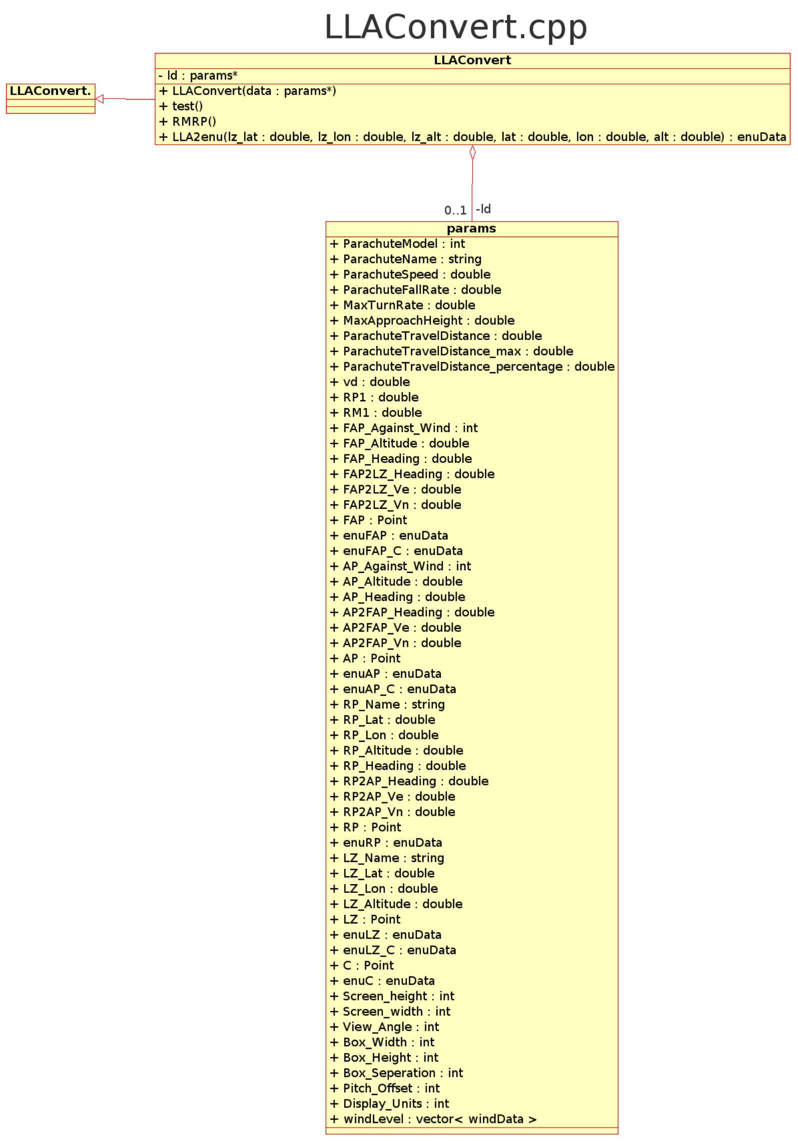


Figure 4.11: LLAConvert class diagram

which is made up of three structures; params, NavParam and enuData. The class uses the data in the static params data to generate the updated ENU positional data.

4.4.2.4 situation.cpp. The situation class is called upon to determine the location of the operator. It then determines which phase of the descent the operator is currently in, whether approaching the Aim Point, the Final Approach Point or the Landing Zone. Once the stage is determined, the appropriate reference and target points are used to calculate where on the path the operator should be located, as described in Paragraph 4.3.2, and Equations 4.13 through 4.15. Figure 4.13 is the class diagram for the situation class. The class structure is composed of two structures, params (See Paragraph 4.5.0.4) and the NavParam (See Paragraph 4.5.0.5).

4.4.3 Display Algorithm Software. The final software components described make up the display algorithm. These components are responsible for utilizing the operator's positional and orientation information and creating the correct situational display. Paragraph 4.3.3 and Equations 4.16 through 4.20 describe the algorithm which is implemented by the following software components.

4.4.3.1 box.cpp. Box.cpp is the class utilized to construct the tunnel display based upon the same fixed location coordinates used in Equations 4.16 through 4.20. This provides an anchor location which is the endpoint for the center of the tunnel. The class diagram for the box class is shown in Figure 4.14, which illustrates the class composition as being made up of two structures, params (See Paragraph 4.5.0.4) and NavParam(See Paragraph 4.5.0.5).

4.4.3.2 draw.cpp. Draw.cpp is used to interface with the SGE graphics library [AND00], used to draw simple shapes, such as lines and ellipses. In addition to the simple shapes, the draw class also implements the textual display utilized to pass information such as operator location and orientation (yaw, pitch, and roll). The draw

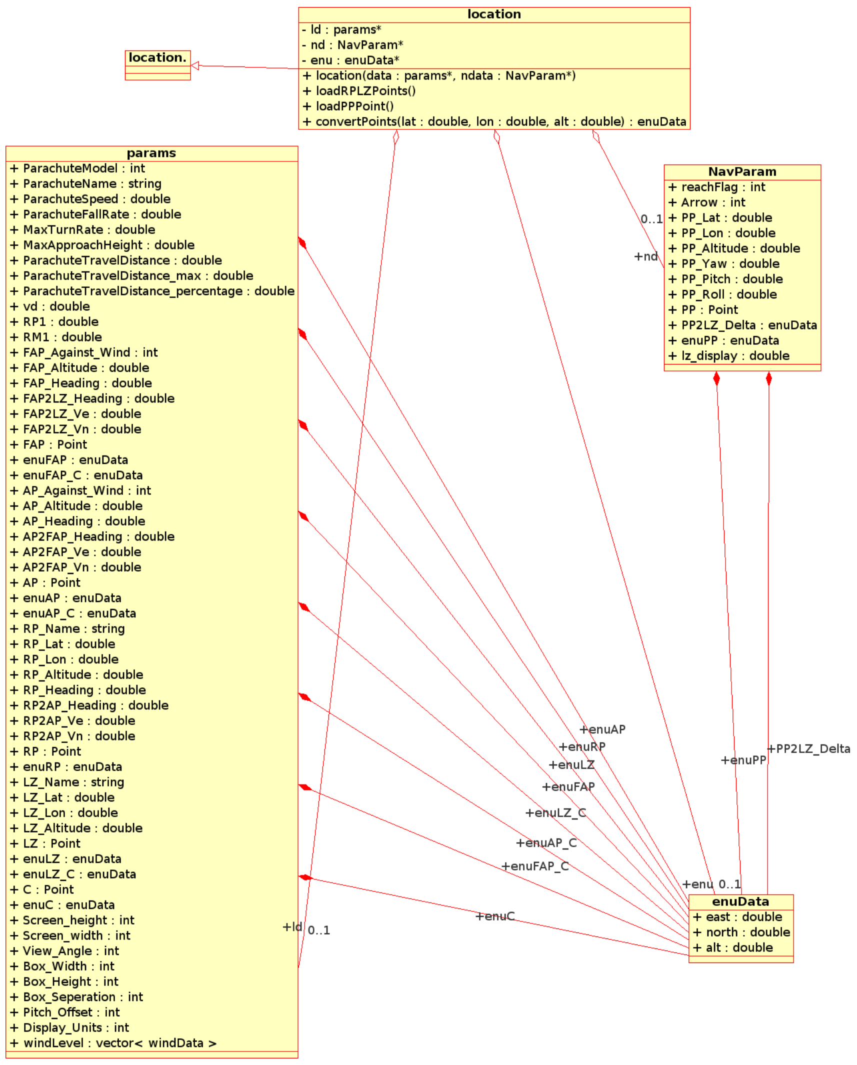


Figure 4.12: The location class reads the present position information from a data structure NavParam (See Paragraph 4.5.0.5) diagram. Location.cpp is made up of three structures; params, NavParams and enuData.

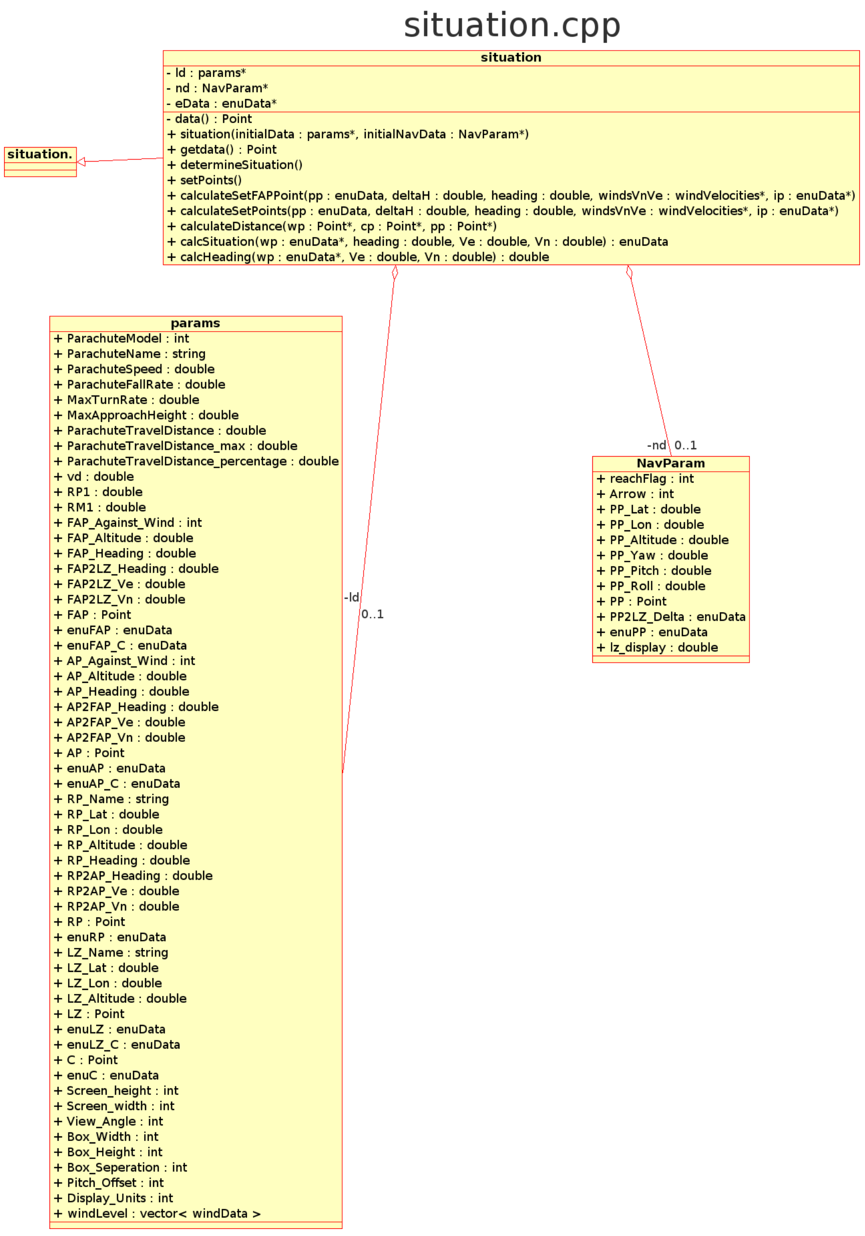


Figure 4.13: The situation class is called upon to determine the location of the operator. It then determines the current descent phase based on positional information. The class structure is composed of two structures, params and the NavParam.

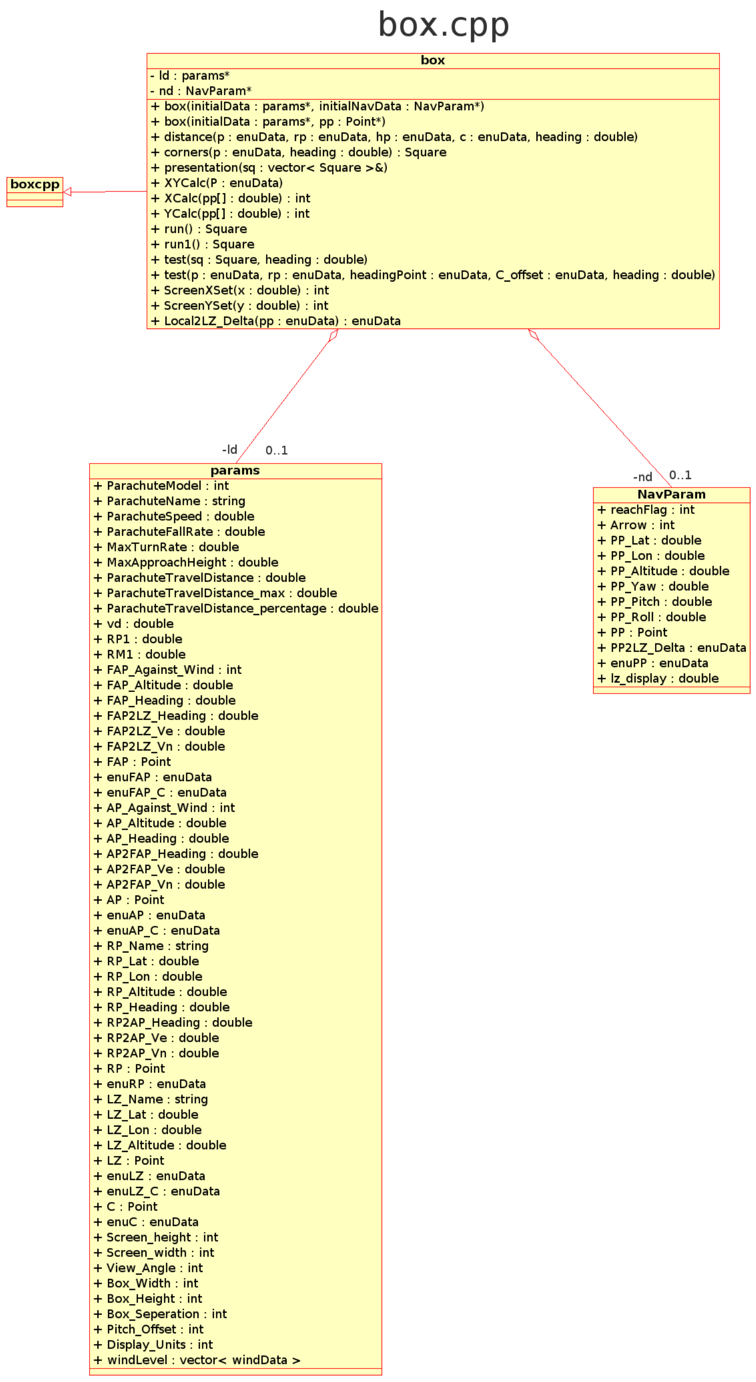


Figure 4.14: Box.cpp is the class utilized to construct the tunnel display based upon the same fixed locations coordinate used in Equations 4.16 through 4.20. The class is composed of two structures, params and NavParam.

methods use pixel locations which have already been rotated into the proper NVG frame and normalized for the screen resolution, as described in paragraph 4.3.3. Figure 4.15 is the class diagram for the draw class, which is composed of two structures: params (See Paragraph 4.5.0.4) and NavParam(See Paragraph 4.5.0.5).

4.4.3.3 path_Projection.cpp. The path_Projection class performs the critical equations used to rotate and normalize the pixel locations, as described in Paragraph 4.3.3 and Equations 4.16 through 4.20. This class converts the physical real-world location into a normalized 2-dimensional pixel location. The display is based on taking a three dimensional physical location and transforming the coordinates into a 2-dimensional screen location. The basis for this transformation is found in [VET]. The program first transforms the ENU Cartesian coordinates into the NVG visual coordinates, as described in Paragraph 4.3.3. Figure 4.16 represents the class diagram for the path_Projection class. This class is composed of two structures: params (See Paragraph 4.5.0.4) and NavParam (See Paragraph 4.5.0.5).

4.5 Data Structures

The software utilizes two main structures, which are the params and NavParam data structures. These two structures are the cornerstones to the navigation program, providing central accessibility for data utilized by other classes. The params structure is used as the model portion of the software; it provides a location where static data is loaded and is accessible by the system. The NavParam data structure provides a location where updated positional information is stored for processing during that particular cycle.

4.5.0.4 params – User Parameters. The user parameters are read by the paramloader (See Paragraph 4.4.1.2, which opens up the config.txt and the navaid.txt files as read-only to prevent data corruption). The user parameters are stored in the params structure, which holds all static jump data, such as the landing zone, release point, final approach, and aim points. Each of these objects in the

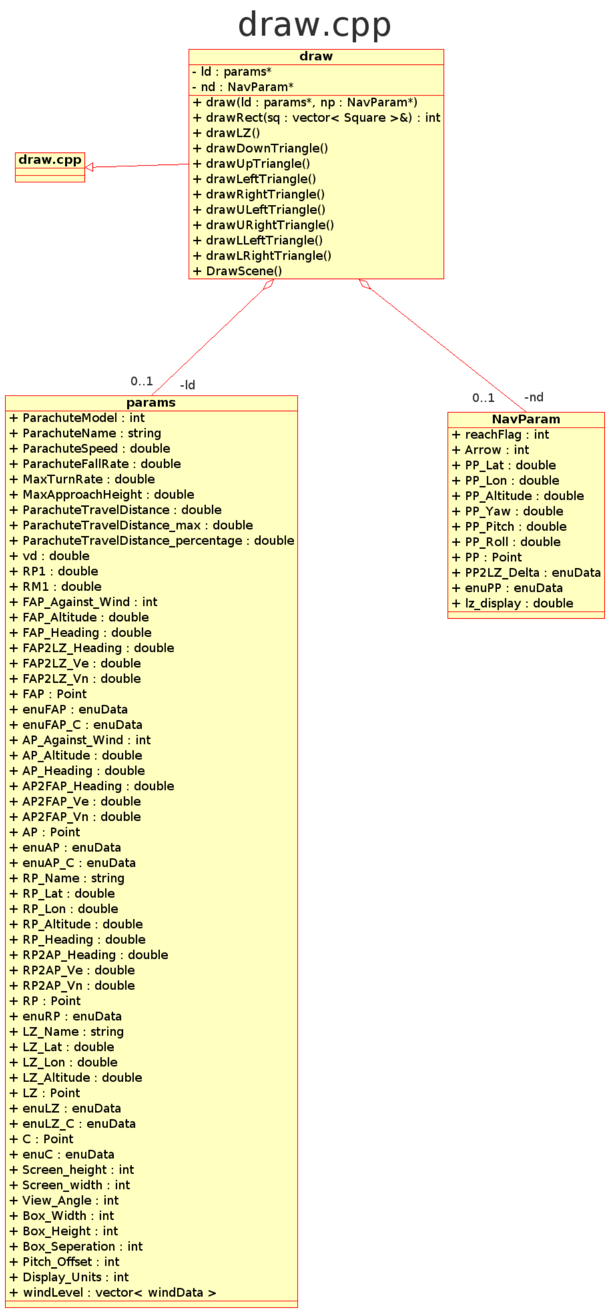


Figure 4.15: The Draw is used to interface with the SGE graphics library [AND00], used to draw the simple shapes such as lines and ellipses. The draw class is composed of two structures: params and NavParam.

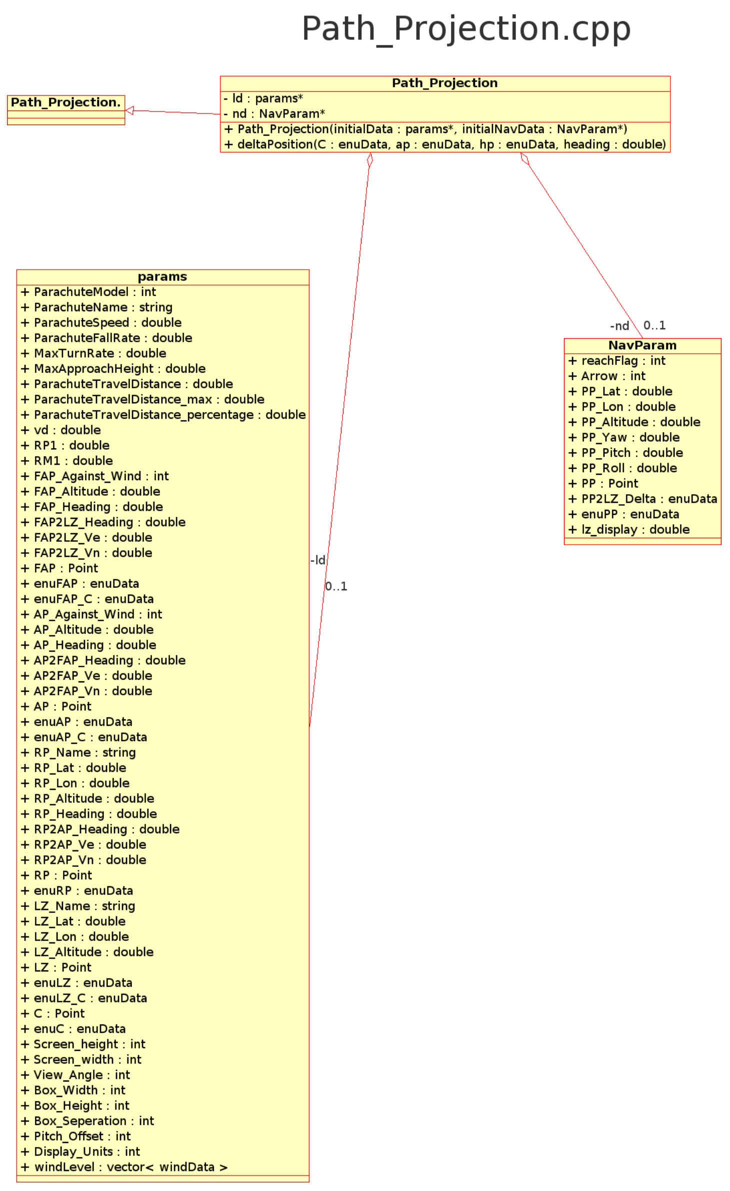


Figure 4.16: The path_Projection class performs the critical equations used to rotate and normalize the pixel locations, as described in Paragraph 4.3.3 and Equations 4.16 through 4.20. This class is composed of two structures: params and NavParam.

data structure has its own information, such as the initialization point, latitude, longitude, altitude, and east/north/up (ENU) location coordinates. A sample of the data structure is located in Appendix A. This structure represents the model portion of the architecture. It provides data which is static and will not change once the initial loading and calculations are performed. This also allows several of the calculations to be performed prior to the mission start (sensor reading cycles). For example, the wind data is contained within this structure. All wind level calculations are performed prior to the sensor reading cycles, which allows the wind data to be stored in a static manner, preventing repeated calculations from being performed. This allows the algorithm to be more efficient since it only performs the calculations once and stores the data for future reference throughout the processing.

There are two structures which are embedded within this structure. The enuData is used to store East/North/Up coordinates for each of the fixed locations. The second structure, windData, is used to store information about each wind level. Each individual wind level (Paragraph 4.4.1.1 and Equation 4.4 describe the process to calculate the wind Data.) is stored as a single windData record within the params structure.

4.5.0.5 navparam–Navigational Positional Data. The NavParam structure was developed to store data for the current report cycle (present position navigational data). This allows all information for a single cycle to be located in a single structure which is standardized throughout the program. The NavParam structure holds present position data, such as latitude, longitude, altitude, and yaw, pitch, and roll orientation.

4.6 Kernel

The kernel is the controlling program within the computer operating system. It is the first part of the operating system to load into memory during booting. It remains there for the duration of the computer operations because it provides services throughout the entire process. Since it remains in memory, it must be as small as possible, but still provide the robustness needed to deliver all necessary services to other parts of the operating system and application programs. Linux was chosen as the operating system, since its kernel could be configured and rebuilt to provide the needed functionality while maintaining the small memory footprint.

The kernel was built utilizing the Kontron Company's toolchain program and scripts. The toolchain consists of a number of components needed to build the kernel. The main one is the compiler itself; the GNU Compiler Collection (GCC), which is used to the host a cross-compiler. This is supported by binutils, which is a set of tools for manipulating binaries. The process used to build the kernel involved modifying the basic build scripts provided by Kontron to remove unnecessary functionality, while maintaining the critical services for the navigational application software and the 80219 processor.

4.7 Root File System – Journal Flash File System 2

A second crucial element of the operating system is the file system structure. The file system provides the structure and organization for storing files and data to allow for ease of access. UNIX and Linux operating systems normally require a graceful shutdown to prevent corruption to the file system structure. The graceful shutdown of this system could not be guaranteed, and in most cases this system will only be shut down by removing power. Because of this reason, the (JFFS2) was selected as the file system to use in this application. This file system was designed to handle embedded Linux operating systems that are not gracefully powered down. It provides this capability by placing the file system directly on the flash memory instead of translating the filesystem for placement on a hard drive.

4.7.1 Software Tools.

4.7.1.1 Eclipse IDE. The primary tool used in the software development of the application software was the Eclipse (IDE). The programming was performed within the Linux version of the Eclipse IDE version 3.2 utilizing the C/C++ plugin. This tool was selected because of familiarity with the development environment and the various tools and plugins available. An additional advantage to this tool is the ability to run under the Linux operating system, which is necessary to perform the cross compilation needed to build the application targeting the XScale architecture. The cross compile capability allows the program to be written, run, and debugged on an x386 architecture then recompiled targeting the new architecture; in this case, the X-Scale ARM processor.

4.7.1.2 Analysis/Design. To enhance the development and analysis of the software, (UML) diagrams are utilized. The Umbrello diagram tool was also utilized to provide a cleaner look at the class relationships. Umbrello works with Linux Eclipse and provides class relationship analysis. The class diagrams used within IV were created using this tool.

4.7.1.3 Software Repository. To ensure the security of the software and the performance of proper backups, a Concurrent Versions System (CVS) repository was established on the AFIT telemark UNIX server. This location is used to keep a historical repository which provides several benefits. The primary benefit being the normal backups performed on the system. By having the data backed up on a regular basis, a functional back up is readily available if something were to happen to the sandbox copy. In addition, to the security provided by redundancy, CVS also provides a historical trail of the updates performed. This allows a working copy to always be available. CVS provides a repository allowing for application rollback if needed. Therefore, if during the course of modifying the software, it is determined that a particular path needs to be reconsidered, CVS provides the avenue to accomplish this task. The final advantage of utilizing the CVS repository is the access to the code from anywhere with an internet connection. This allows for work to be accomplished on a current version of the software without dealing with synchronization issues.

4.7.2 Software Development Process. The iterative development method was used for this project to benefit from the stability offered. In addition, the software development approach was also modelled after the Microsoft cooperation, by not allowing branches (target and development) to diverge and by also maintaining a functional code. Backups were performed at least once daily during the development process and were always functional. This means the backups were always in a state which could be compiled and run. The backup would not contain all the functionality and many of the functions were stubbed out, but the software was in a stable state which could be run.

By utilizing this type of backup system, the software was also constructed utilizing several incremental builds. The iterative development process was used to break the project down into small iterations and provide a stable foundation for future growth. For example, the first phase of the project developed the parsing and data file loading classes, then the wind algorithm classes were developed. This allowed the system to be grown while still maintaining an operational system. The multiple smaller builds avoided the waterfall method and provided a logical breakdown of the entire system. This type of development also allows testing and bug resolution prior to the final stage of development. The iterative process has multiple cycles of planning, requirements, analysis & design, implementation, testing, and evaluation.

There were two distinct development branches utilized for this project. The first was the development system, which was an Intel architecture Dell laptop running an Intel core Duo, with the Linux operating system. This system was chosen as the development system because of its robust debugging and multi tasking capability. This is a deviation from Microsoft, in that development was not performed on the actual system which the application was intended to run upon. The second development branch was the cross compile targeted XScale processor. This computer processing board that runs during the operator descent provides the navigational path and display processing. Once the software was written and in a stable state, a cross compilation was also executed. This entailed ensuring the libraries and header files were all properly compiled to run on the different architecture. By keeping both branches synchronized, all debugging and compilation issues were resolved throughout the development, instead of waiting until the last iteration of development.

4.7.3 Design Patterns.

4.7.3.1 Model-View-Controller Pattern. The application software is implemented following a Model-View-Controller pattern (MVC). The MVC divides and redistributes responsibility so that classes and packages stay small enough to maintain [MET03]. In this project the division is aligned with the three algorithms described in Chapter IV. The model portion of the architecture is the wind algorithm, (See Paragraph 4.3.1), which defines the static data used throughout the software. Once the program begins, there are no updates to the parachute, wind or user parameters. The controller portion of the system is the CalcSituation algorithm (See Paragraph 4.3.2), which is the updated information based on present positional data. This controls the system application. The display algorithm, (See Paragraph 4.3.3), represents the view portion of the system.

4.7.3.2 Interface Pattern. The next pattern implemented is the interface pattern. All classes have a header file which defines the interface needed to interact with methods and data in a given class. A benefit of an interface pattern is it imposes a limit on object interaction. The limitation allows for ease of maintainability by allowing the class which implements an interface to be changed in how it fulfills the interface, while leaving the client unaffected [MET03].

4.7.4 UML Analysis.

4.7.4.1 Class Diagram. Class diagrams are one of the primary tools of object-oriented analysis and design. They show the classes of a system, their interrelationships, (including inheritance, aggregation, and association), and the operations and attributes of the classes. Class diagrams are used for a wide variety of purposes, including both conceptual/domain modelling and detailed design modelling. In this project class diagrams are shown in section 4.4, which describes the various classes which make up the system algorithms and their relationships.

4.7.4.2 Use Case Analysis. Use case diagrams provide an overview of the usage requirements for a system. They are useful for presenting the requirements in a simple straight-forward manner. Use cases describe "the meat" of the actual requirements. The use cases listed below were used in determining the software flow for various jumper situations. These cases show what will occur when the jumper is on the average glide slope (best case), when the jumper is at the maximum glide slope to the target, when the jumper is beyond the maximum glide slope to the target, and when the jumper is in danger of over shooting the target. Figure 4.3 shows the possible jumper locations and how these relate to the four cases developed below.

Use Case 1: Jumper on average glide slope

Scope: CalcSitution algorithm, Display Algorithm

Level: subfunction

Primary Actor: Jumper

Precondition: Parachute parameters have been input and wind calculations have been performed

Main Success Scenario:

1. Jumper is following navigational data
2. Jumper is on glide slope
3. "On Track" Message is posted to HUD

4. Wind data is accurate
5. Jumper is able to reach aim point with average glide slope

Extensions:

1. Jumper remains on glide slope, no change needed
2. Winds remain stable

Technology and Data Variations List: None

Open Issues:None

Use Case 2: Jumper is at maximum glide slope

Level: CalcSitution algorithm, Display Algorithm

Primary Actor: Jumper

Precondition: Parachute parameters have been input and wind calculations have been performed

Scenario:

1. Jumper is at the maximum glide slope distance
2. "Max Glide Slope Needed" Message is posted to HUD
3. Tunnel display is updated according to Equations 4.13 through 4.15, to reflect course deviation
4. Jumper uses maximum glide to adjust position
5. "On Track" Message is posted to HUD
6. Jumper reaches aim point

Extensions:

1. Jumper does not adjust to navigation messages
2. Tunnel display continues to show position off of intended navigational path – Path does not recalculate to new position

3. Jumper continues to drift out of navigational parameters – See use case 3 “Beyond glide slope”

Technology and Data Variations List: None

Open Issues: None

Use Case 3: Jumper beyond maximum glide slope

Scope: CalcSitution algorithm, Display Algorithm

Level: subfunction

Primary Actor: Jumper

Precondition: Parachute parameters have been input and wind calculations have been performed

Scenario:

1. Jumper is beyond maximum parachute travel distance
2. “Beyond Reach” message is posted to HUD
3. CalcSituation recalculates tunnel display parameters
4. New navigational data is updated
5. “On Track” message posted to HUD
6. Jumper reaches aim point

Extensions:

1. Jumper’s position is out of reach of final aim point
2. “Miss Target” message posted to HUD
3. Navigational tunnel still shows original path

Technology and Data Variations List: None

Open Issues: None

Use Case 4: Jumper overshoot possibility

Scope: CalcSitution algorithm, Display Algorithm

Level: subfunction

Primary Actor: Jumper

Precondition: Parachute parameters have been input and wind calculations have been performed

Scenario:

1. Jumper is within the minimal glide radius
2. "Over Shoot Warning" Message is posted to HUD
3. Jumper bleeds off excess altitude
4. Tunnel display continues to show position off of intended navigational path – Path does not recalculate to new position
5. Jumper's altitude is readjusted to navigational data
6. "On Track" message posted to HUD
7. Jumper reaches aim point

Extensions:

1. Jumper does not adjust flight path
2. "Miss Target" message posted to HUD
3. Navigational tunnel still shows original path

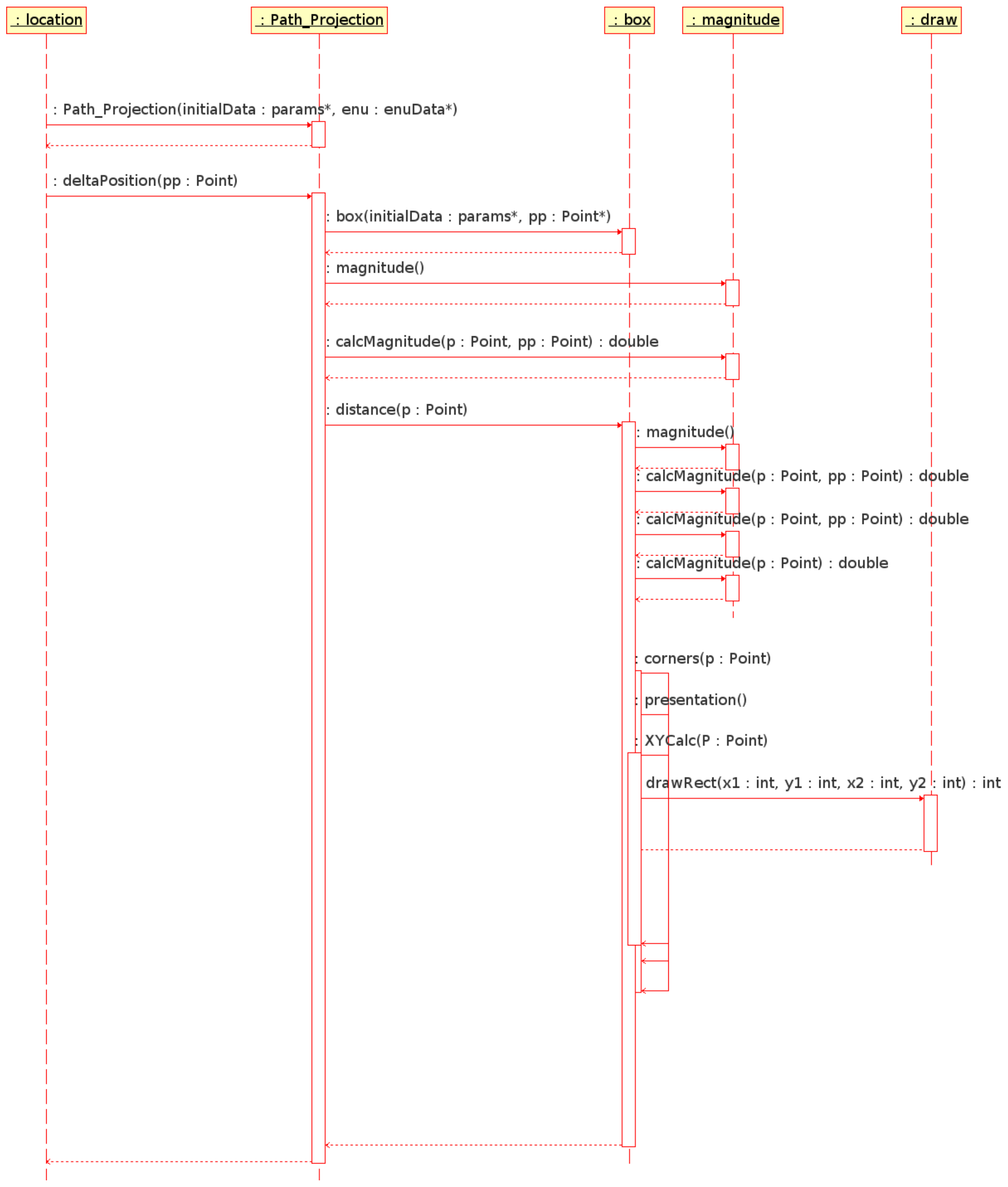
Technology and Data Variations List: None

Open Issues: None

4.7.4.3 Sequence Diagram. UML sequence diagrams model the flow of logic within a system in a visual manner, enabling both documentation and validation of logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behavior within a system [MET03]. Figure 4.17 is the sequence which shows the location class flow. The diagram describes the system flow from the location class. This class will call the path_projection class, which is used to calculate the proper pixel display location, as described in Equations 4.13 through 4.15, by calling classes such as box.cpp and magnitude.cpp, which finally calls draw.cpp to present the data.

4.8 Summary

Chapter four described the three algorithms which make up the developed application. Further, the various software components which were developed to implement the various algorithms were also described. The software is also depicted by utilizing class diagrams, which show the relationship between the various classes and the data structures. The model/view/controller architecture used to model the software is also related to the three algorithms and their interrelationship. The final portion of this chapter briefly describes the software practices and tools used to aid the software development.



: location
: Path_Projection
: box
: magnitude
: draw
: Path_Projection(initialData : params*, enu : enuData*)
: deltaPosition(pp : Point)
: box(initialData : params*, pp : Point*)
: magnitude()
: calcMagnitude(p : Point, pp : Point) : double
: distance(p : Point)
: magnitude()
: calcMagnitude(p : Point, pp : Point) : double
: calcMagnitude(p : Point, pp : Point) : double
: calcMagnitude(p : Point) : double
: corners(p : Point)
: presentation()
: XYCalc(P : Point)
drawRect(x1 : int, y1 : int, x2 : int, y2 : int) : int

Figure 4.17: The location class sequence which shows the program flow. Location.cpp calls the path_projection, class which is used to calculate the proper pixel display location, as described in Equations 4.13 through 4.15, by calling classes such as box.cpp and magnitude.cpp, which finally calls draw.cpp to present the data.

V. Methodology and Results

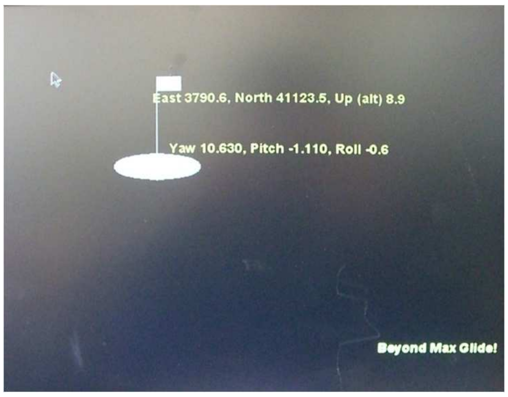
This chapter discusses the techniques used to evaluate the graphical HUD developed for this project and its impact on the operator's ability to track a designated target location. The ability to track a target location involved both the ability to keep a target within the display screen and the ability to reacquire the target after looking away by utilizing directional arrows. This evaluation was conducted on the ground for safety and cost reasons, therefore the main point of evaluation was the amount of lateral (yaw) head movement needed to track the target. Pitch and roll head movement was not as critical during this evaluation, since altitude was not a factor. This test does not test the entire system. For example, the tunnel in the sky approach was not able used in the ground testing. There were plans to accomplish additional tests beyond what are shown here, but due to time constraints these were not able to be accomplished. The point of the testing described in this section is to demonstrate that the system is able to provide visual cues which enable a user to look in the correct direction (towards a fixed target). This capability is fundamental to the more advanced display algorithms described in Chapter 4.

5.1 Goals and Hypothesis

The goal of this research was to design a navigational aid to assist HAHO jumpers reach their desired location in darkness or inclement weather. This research broke this goal into two portions, each of which will be addressed separately; first the head tracking algorithms and second, the impact of the navigational aids in reaching a target destination.

Since work had been done prior to this research in determining which navigational display would provide the more intuitive reading for the individual, this research will not address that particular issue, but will instead focus on the ability to navigate and reach a destination at various distances.

The navigational HUD display is an improvement of the system utilized to test the prior system by Thompson [THO05]. The NVG system utilizes an E-HUD system,


East 3790.6, North 41123.5, Up (alt) 8.9

Yaw 10.630, Pitch -1.110, Roll -0.6

Beyond Max Glide!

Figure 5.1: Screen shot of the HUD display.

boasting a 46 degree field of view, which is a vast improvement over the previous 13 degrees. Another major advancement this project produced was the addition of a head tracking system, which is utilized to monitor the operator's visual field.

The navigational display shows a small flag centered upon an oval to clearly identify the targeted landing zone. If the LZ is not within the field of view, then directional arrows will indicate the appropriate vector needed to reacquire the target. The tunnel in the sky is represented by various numbers of squares, leading to the landing zone, dependant upon the distance to the LZ. The textual display is located across the top and bottom of the display. The indications are the east, north, and up distance to the target, with the next line down indicating the operator's orientation in yaw, pitch and roll of the helmet mount sensor. Across the bottom is a heading display (yaw information) and target distance information (See Figure 5.1).

5.2 Approach

The research goal was to evaluate the effects of the added navigation technology to the current night vision goggle capabilities. Due to the danger and expense involved in actually performing an airborne jump, ground testing was conducted. This provided a first look opportunity at the software and algorithms used to both track head movements and identify targeted locations. This is a crucial first step to ensure all hardware components operate as expected–providing the necessary data in the expected time frame.

The first portion of the test was designed to test the quality of the head tracker in both stationary and moving scenarios. Head tracking allows the user to move his head or be in motion while not looking directly at the target and still able to reacquire the target in the correct position. The second portion of the test will evaluate the user's ability to utilize the navigational aides to effectively reach a predefined location at various distances.

5.3 System Description

The components that made up the system under test include:

- GPS system–The system is made up of the various components and software described in Chapters 3 and 4.
- Test subject– Individual who was responsible for acting upon the supplied navigational information.
- Test monitor/safety–Individual responsible for monitoring the test and ensuring hazards were avoided.
- Dell Latitude (2 gigahertz, 512 megabytes of RAM, running a standard AFIT ghost XP configuration) laptop–used to run software test software under a controlled computing environment.
- STS LPNVG PVS/AV-21 Monocle, with the standard E-HUD.
- Dell Inspiron E1505 Core Duo–Utilized for charting the test course.
- Garmin Legend e-trex–Handheld GPS receiver, used as baseline test.

5.4 Evaluation Technique

The ideal evaluation is to have an experienced HAHO jumper utilize the system and evaluate the effects of the navigational aides. There are several test jump organizations who are willing to actually perform this test. For example, one of the primary sponsors of the research was the Army's Research and Development Center in Natick, MA, who performs these types of experimental jumps with various navigational equipment. Unfortunately, without further verification testing and evaluation of the system, this is an unsafe option. Test jumpers would be risking their lives while evaluating the system, and therefore the system must have a high degree of confidence which has not yet been obtained.

The second option, which is less dangerous, is to utilize simulators and emulators. A parachute simulator maintained by the Ohio National Guard was evaluated for testing purposes, but unfortunately was found to be unsuitable for the evaluation of this system. The simulator places the individual in a parachute harness, and has the operator wear a set of virtual reality goggles, and simulates the movement which occurs in various weather and altitudes. The use of the virtual reality goggles eliminates the ability to utilize this system as a viable test environment since it does not allow for information to be fed into the GPS/head tracking system developed.

The simulation originally developed by Balaz provides the movement simulation and the tunnel display upon the desktop for a safe environment, but does not have an interface to the NVG system nor does it provide a simulation for the newly developed head tracking system. Therefore, neither of these simulations could be utilized for the evaluation.

Due to time constraints, a realistic simulation was not developed for this system. This would be the logical next step, developing a system which not only emulates winds upon the path, but also accounts for the head movements of the user.

The final option chosen was the ground evaluation of the system. This option also had several constraints, but allowed the system to be evaluated by tracking movements of the test individual in a safe, controlled environment. The system was developed for use by HAHO jumpers and therefore, also utilizes altitude within the navigation algorithm. Unfortunately, by performing ground testing, the 3rd dimension was removed from the equation. In addition, the winds were also removed from the equation due to the fact that a ground walker is not affected by the winds in the same manner as a parachutist, during freefall.

5.5 *Experimental Design*

The test was designed to test two distinct aspects of the research. First the head tracking algorithm was tested. This evaluation was designed to evaluate the accuracy of the head tracking system, which was a newly implemented development. Previous work has been conducted on the "best" navigational message to display, but none of these had the ability to track the user's head movements and incorporate that into the display. This evaluation also addressed possible issues with the MIDG internal magnetometer.

The first portion of the test was designed to test the internal magnetometer and the effect that various environments could have upon its operation. The test placed the unit in a stationary position, allowing it to track a target over a period of time to record the amount of drift noted. The test was conducted both within a build with an external GPS antenna feed and in an outside field free from obstacles and at least 100 meters from power lines and light poles. The amount of drift was tracked and noted.

The second test was designed to evaluate the ability to navigate to a location while in a moving vehicle. A path to three different points was plotted and used as the predefined coarse. This test was designed to determine how accurately and efficiently the paths could be followed utilizing the various systems. The differences were noted in both the head movement and the amount of time the target was off screen during navigation to the target location.

5.6 Workload

The workload submitted to the system consisted of the distance between the origin and the target. The distance between tunnel squares and the size of the squares was kept at a constant, and therefore did not influence the test operation. The distances ranged from several meters to over a kilometer. In addition, various paths types were chosen, such as in an open field or around buildings.

5.7 Performance Metrics

The primary performance metric utilized by the evaluation was the amount of head movement used to navigate to the target LZ. This head movement had a direct effect on the time needed to operator needed to reach the designated target. In addition, the time to reach each target was also used as a performance metric to determine the effectiveness of the navigational aids. It is assumed that the more intuitive and responsive the display, the quicker the user will be able to follow the navigational data.

5.8 Test Parameters

The following parameters were factors in the ground testing of the system:

- Environmental conditions–This includes the air temperature, precipitation levels and the terrain being navigated.
- Lighting Conditions–The system is designed to be utilized during hours of darkness.
- Individual–Different users have different abilities to understand and follow navigational data.
- Terrain–The course traversed various types of terrain, such as paved pathways, grassy areas and hilly terrain.
- GPS accuracy.

5.9 Suggestions for Future Testing

Ground testing was utilized because this was the safer option, however this method had several limitations which affected its actual relation to a real world parachute jump. To build confidence in the system and identify all potential software errors and hardware limitations, the system must be fully stressed by lab modelling.

The next step would be to develop a simulation which can simulate winds, descent rates, and environmental influences. The simulation must then be able to read inputs, such as head tracking and user steering inputs, and incorporate these into a realistic scenario, which will stress the system. The navigational messages output by the simulation must be realistic and correctly direct the operator to the intended landing zone.

Once confidence in the system has been established via extensive lab modelling, the final step will be to actually have the system jump tested by qualified parachutists. The safety precautions must be explored and developed to account for system failure. The operator must have the ability to navigate and safely land if the system completely fails. A second consideration is to have a control system jumped at the same time to ensure the system does not output erroneous navigational data which could lead the jumper into a hazardous condition, such as sending him off course into a populated area or a water landing. Both of these scenarios could expose the jumper to undo danger.

A final consideration prior to having the system actually jump tested is to ensure the hardware components meet the safety requirements needed. For example, the system must be sealed to ensure moisture does not adversely affect the operation. Video and input cables must be safely secured to not interfere with parachute operations, causing a hazard by fowling the chute deployment.

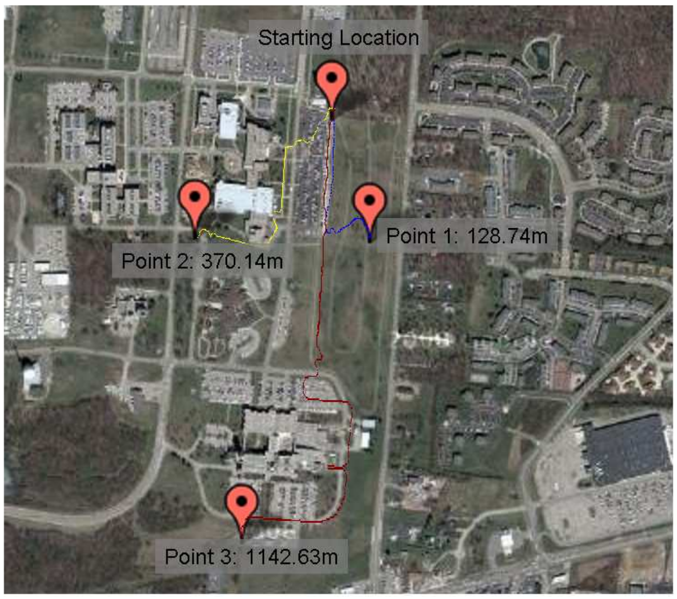


Figure 5.2: Map of locations, distances and paths followed during the evaluation.

5.10 Test Design

5.10.1 Head Tracking. The test was designed to test two aspects of the implemented research; the head tracking and the navigational impact of the additional information. The head tracking aspect of the test did not address the tunnel or navigational capabilities, but only the ability of the operator to correctly track and reacquire a target at various distances. Although the additional navigational data, such as the tunnel in the sky, were available and operational during the test, this test focused on the use of directional arrows to indicate correct look vectors.

The baseline used for this test was to mount the head tracking unit in a fixed forward facing position and follow the course to the intended target location. This recorded the amount of variance the target was actually off screen. This test was repeated to all three different locations. The test was then repeated while having the golf cart follow the predefined path, but while having the test subject try to keep the target within the visual screen. This test was also conducted to all three designated target locations.

5.10.2 Path Navigation. The second portion of the test utilized both the head tracking and the tunnel in the sky concept to navigate to a designated location (See Figure 5.2). The predefined course presented obstacles which caused the system to adjust its navigational solution with a new course. Due to a last minute failure of one of the body worn display components, the test was modified from its original configuration. The test was run on the ground utilizing a laptop to provide the display interface. This did have an impact on the test due to the increased speed and memory of the laptop as compared to the body worn computer design. To provide the necessary power to run the head tracker and the laptop, the test was conducted on a golf cart with an onboard power supply. The maximum speed utilized during the test was 10 mph, with an average speed of only 6.8. The use of the golf cart also presented an influence on the test and outcomes, since now timing was not effected by the individuals walking, but instead on the ability to pass navigational data to the cart operator.

Again, since this was a ground test, neither wind nor altitude were significant factors in the overall test. This was very apparent by the small amount of variation in the pitch recorded during the various test runs. It is speculated that during an actual parachute decent, the pitch and roll aspects of the head movement would be far more drastic.

5.11 Test Results

5.11.1 Head Tracking. The first set of test reinforced the initial hypothesis that attitude would not be a factor while performing ground test. This was very apparent by the tracking of the head movements. Figure 5.3 show three distinct tracks for the yaw, pitch and roll. During all test runs, the pitch and roll were only slightly affected, while extreme fluctuations were noted in the yaw during the same time period. This same data is also shown in Figure 5.3 with the yaw value adjusted to show only positive values. If a value was negative, 360 would be added to value, in order to bring the range between 0 and 360 degrees.

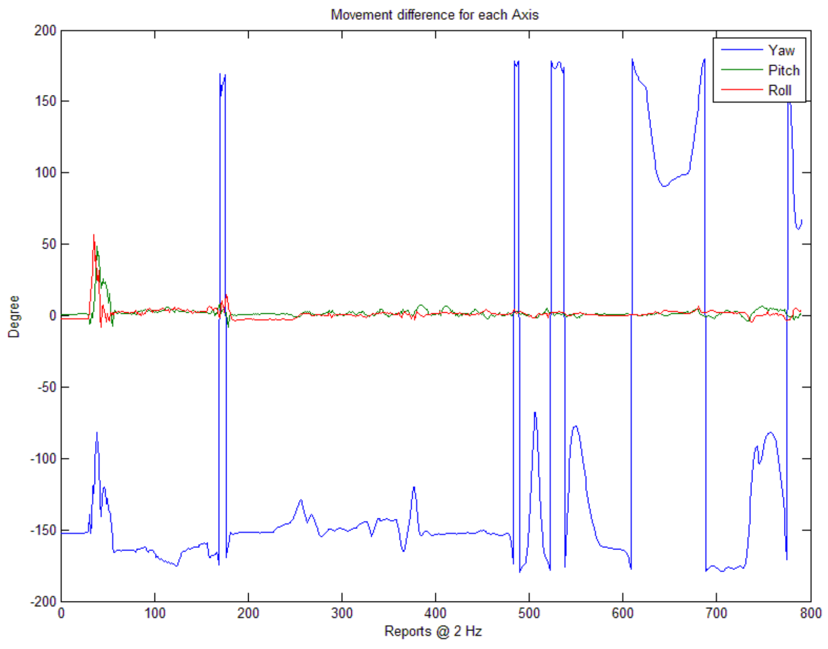


Figure 5.3: Plot showing slight movement in pitch and roll, but large amount of movement needed in the yaw axis to keep target within screen.

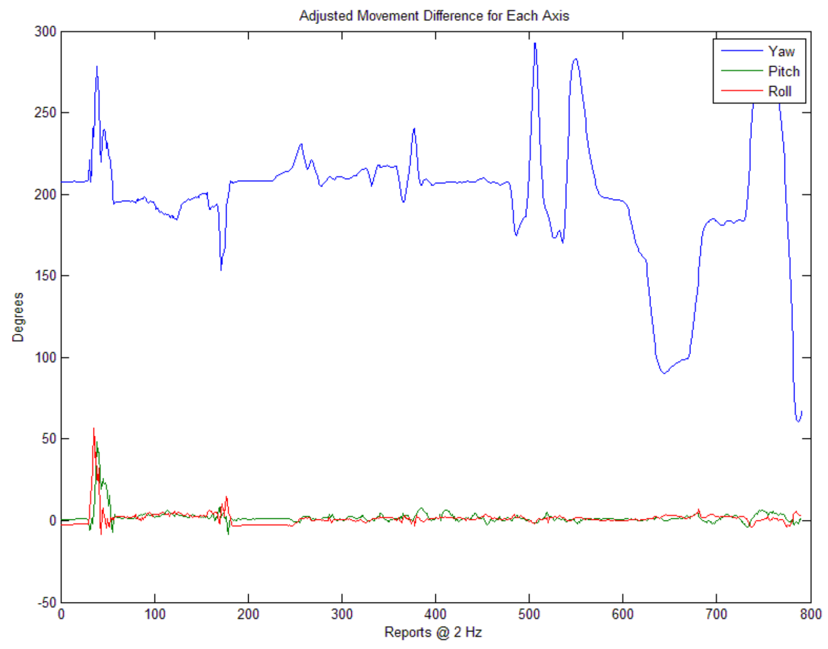


Figure 5.4: This plot uses the same data as Figure 5.3, but the data data has been adjusted to show the ranges between 0 and 360.

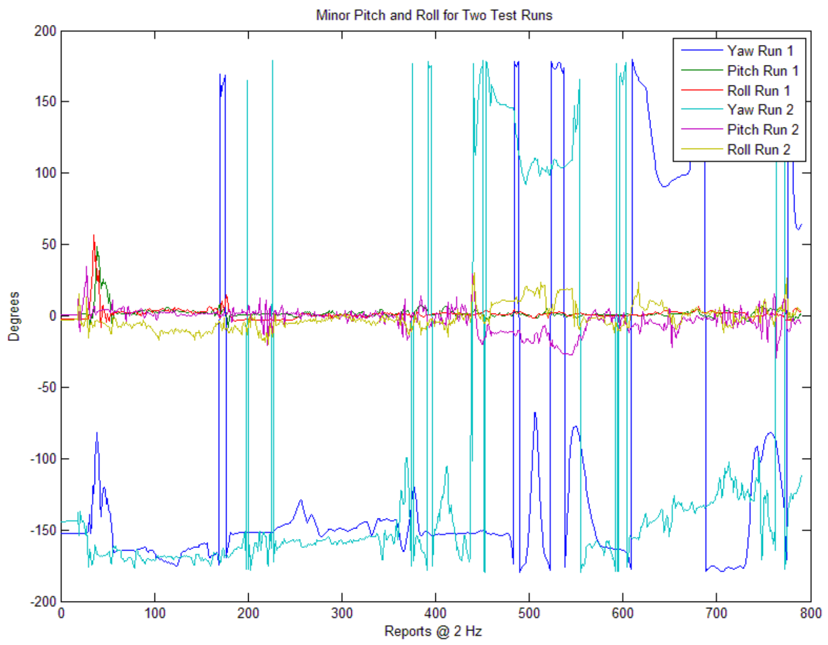


Figure 5.5: Two test run tracks which shows extreme movements of the yaw axis in comparison to pitch and roll.

Figure 5.5 shows the pitch and roll fluctuations for 2 test runs to point 2 (370.14 meters). This shows that the altitude portion of the navigation algorithm had very little influence on the ground test. Since it is speculated that airborne personnel would have a much greater variance and use of all three axis; yaw, pitch and roll, a simulation must be created to allow the operator to exercise all three axis. Because the yaw movement can have a 360 degree change, Figure 5.6 is the same plot used in Figure 5.5, but adjusted for positive values only. If the degree was negative, 360 was added to the value in order to bring all values between 0 and 360 degrees.

During the testing of the head tracking, the amount of head movement by test subjects at times caused the head tracker to become disoriented and became 180 degrees inverted. This is believed to be due to an issue with the internal magnetometer. Figure 5.7 shows the amount of head (yaw) movement for two test runs. The first run (blue) line is a stationary mounting of the head tracking unit on the golf cart following the predesignated path. The objective of this test was only to see how much variation

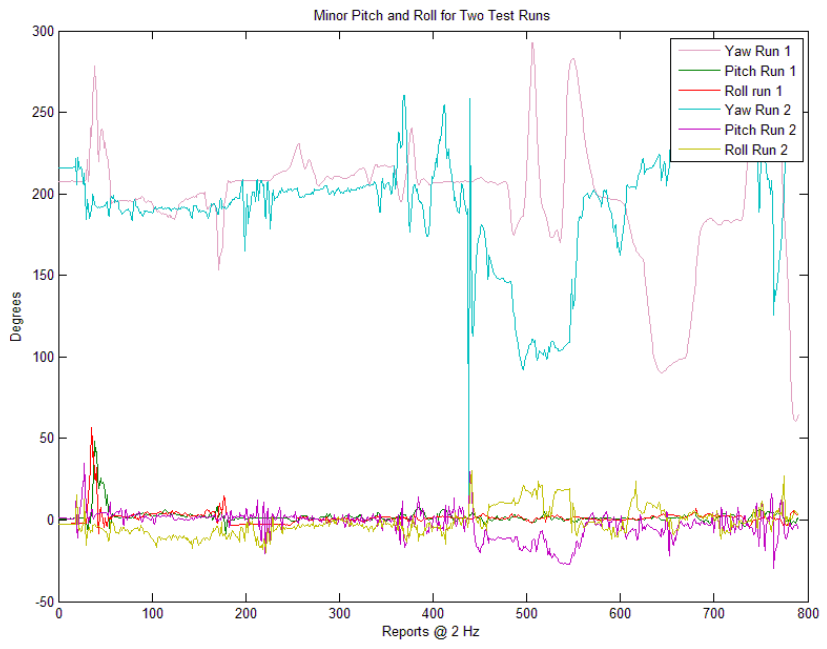


Figure 5.6: Same test run track data shown in Figure 5.5. This plot has yaw 360 degree adjustment in order to bring the values between 0 and 360 degrees.

there was in the yaw movement, without attempting to keep the target within the operator's target. This was following a path, without any navigational assistance from the system. The spikes in the the yaw are due to the turns in the path which caused the momentary spike in the degree of movement. The second line (red) represents the amount of movement (yaw) of operator's head while attempting to keep the target in the screen while in a moving vehicle (golf cart). The data shows that an operator's head must be in almost constant movement to adjust for the target tracking. The minor corrections allowed the operator to keep the target within the screen, but it also indicates that the smaller movements were also due to the effectiveness of the directional arrows. The efficient use of the vectoring arrows allowed the operator to make minor adjustments to the look vector to reacquire the target. This was a successful test of the head tracker with no head tracker disorientation.

Figure 5.8 is the yaw tracking data for a test run conducted to a target location which was 1142 meters from the starting location. This plot shows the entire run,

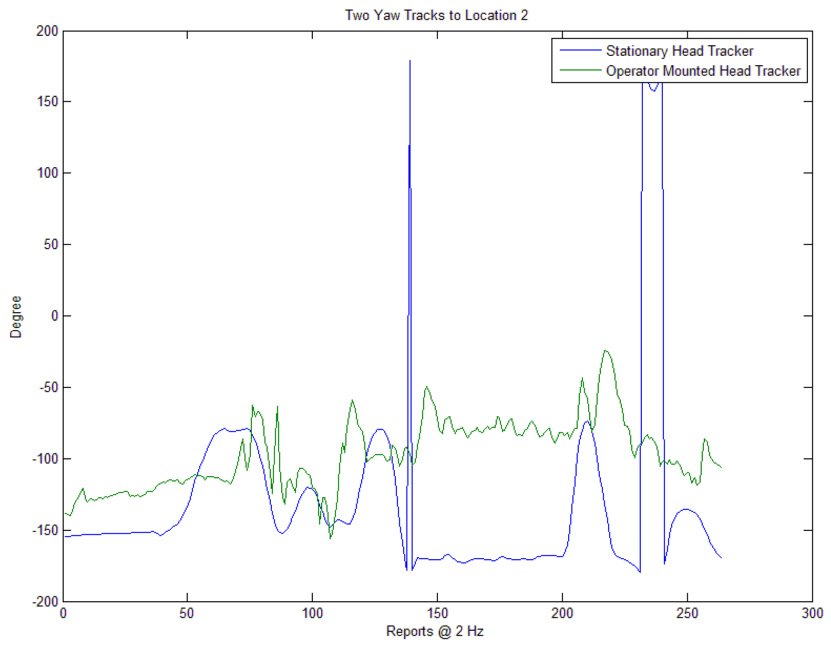


Figure 5.7: Forward facing mounted head tracker (previous research), operator mounted head tracker. Amount of movement noted in attempting to keep track within screen.

but is intended to highlight how the test subject became disorientated at about the 4 minute mark. The yaw track became almost 180 degrees off normal tracking and much more movement in the pitch and roll axis appears, as the operator attempts to reacquire the target. Figure 5.10 represents 5 minutes of the run, intended to show the effects of the head tracking disorientation. During this period, for about a minute period the operator was tracking data at 180 degrees off target. The line follows a fairly smooth track for the first four minutes with only momentary spikes in the yaw, but then the head tracker becomes disoriented at about three and half minutes into the reading. The tracking average moves from negative 150 degrees to positive 150 area of tracking. Figure 5.9 is the same data track adjusted for yaw's 360 degree offset. If a report was negative, 360 was added to bring the report back into the positive set. The disorientation period is still visible at the same time. As a note, the time graph was truncated to 600 reports for clarity of information. During the test, the safety

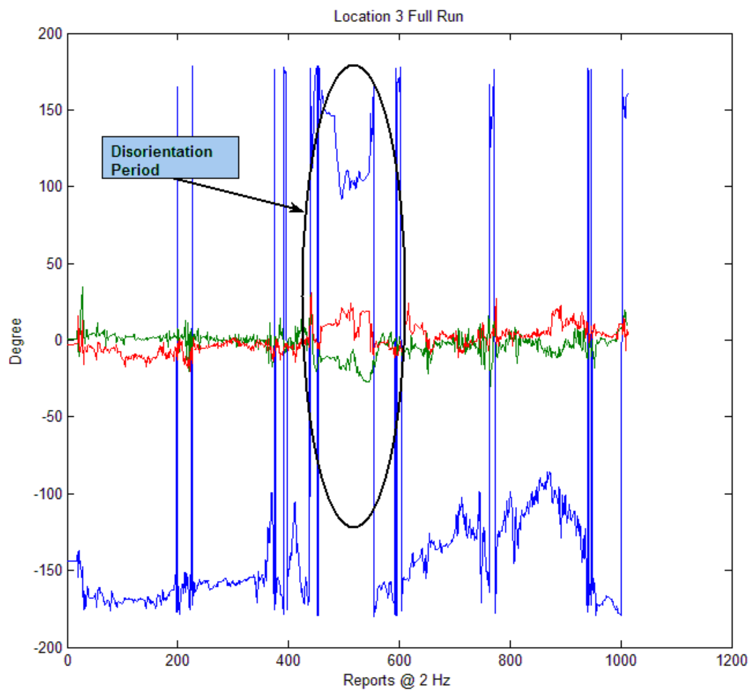


Figure 5.8: Data tracks for the 2nd run to location 3. This Data shows how the operator becomes disoriented, affecting all three axis as he attempts to reacquire the target location.

monitor intervened to avoid any possible safety issues and returned the test subject to tracking the orientation for the remainder of the test.

The plot in Figure 5.11 illustrates the difference between the heading angle to the target and the operator's head tracking yaw. The blue track indicates the heading angle to the target. The green plot indicates the human yaw tracking which took place in order to keep the target within the screen. The difference between the two plots indicate the amount of movement needed by the individual in order to keep the target within the screen.

Fiqure 5.12 represents the amount of drift and correction which is imposed upon the head tracker with no movement. The data represent 58 minutes worth of tracking, where the target was acquired and the system was then set on a stationary workbench. This system was connected to an external GPS antenna, but the head tracker was

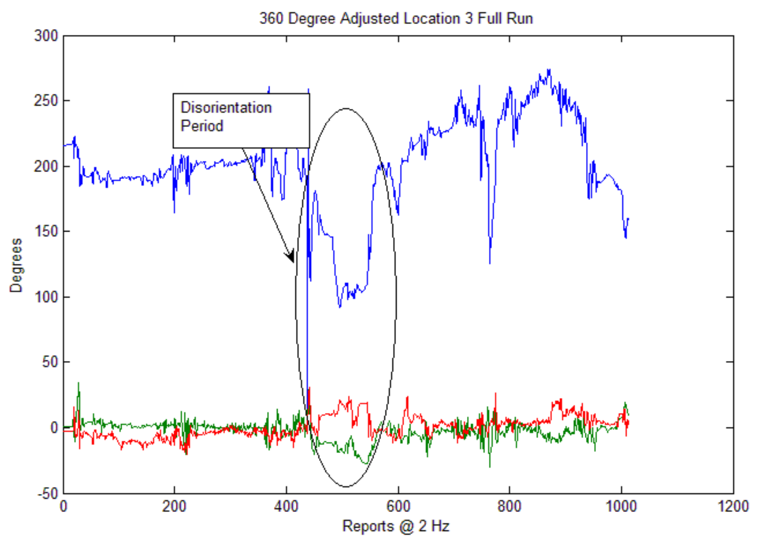


Figure 5.9: Data tracks for the 2nd run to location 3, this is the same data used for 5.8. The data track has been adjusted for yaw's 360 degree offset. If a report was negative, 360 was added to added to the report, to make it positive.

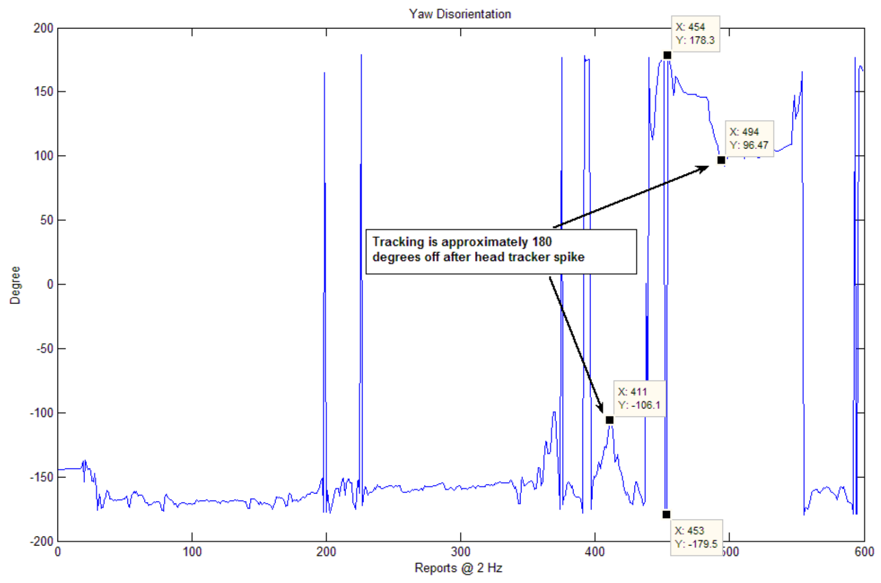


Figure 5.10: Yaw data for the test run to location 3. This data indicates a 180 difference from report 453 to report 454. After this point, tracking is almost 180 degrees different from previous target tracking.

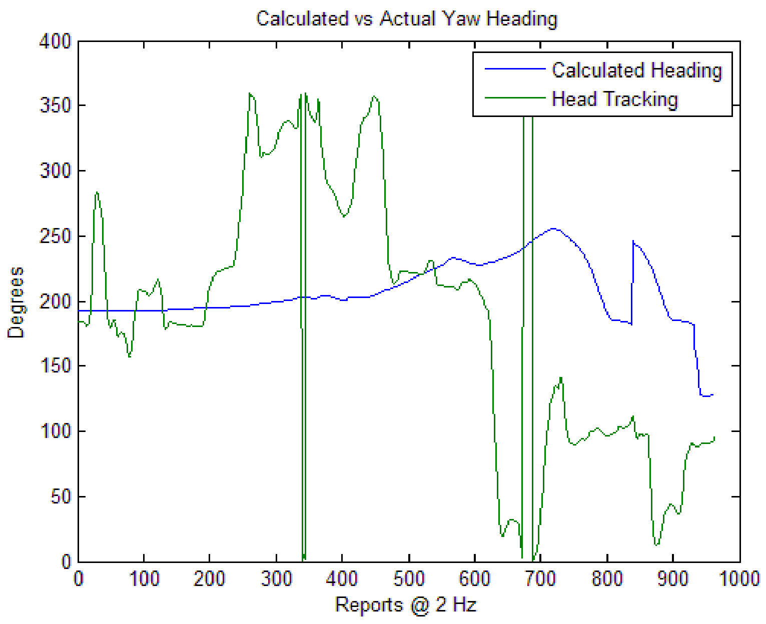


Figure 5.11: This plot illustrates the difference between the heading angle to the target and the operator's head tracking yaw. The blue track indicates the heading angle to the target as calculated by the software. The green plot indicates the human yaw tracking which took place in order to keep the target within the screen.

surrounded by electrical equipment to induce a high average electrical field. The electrical interference could have been both much worse (placing strong magnetics near magnetometer) or much less by having the system placed in an open field for the period. This test shows under somewhat normal operating conditions, the stationary drift of almost 2 degrees of yaw for the hour period is uncorrected. Although, when in a moving vehicle without dramatic changes in direction, the tracker seems to hold the same type of tracking. The disorientation while in movement, problem seems to only occur when drastic (over 90 degrees) changes in direction are made. This is only preliminary speculation, and would require additional testing, focused solely on the movement effects, to determine the true effects.

5.11.2 Path Navigation. This test was initially intended to utilize time as the primary metric to measure the difference between the various types of navigational aids, in addition to provide some comparison to previous research work.

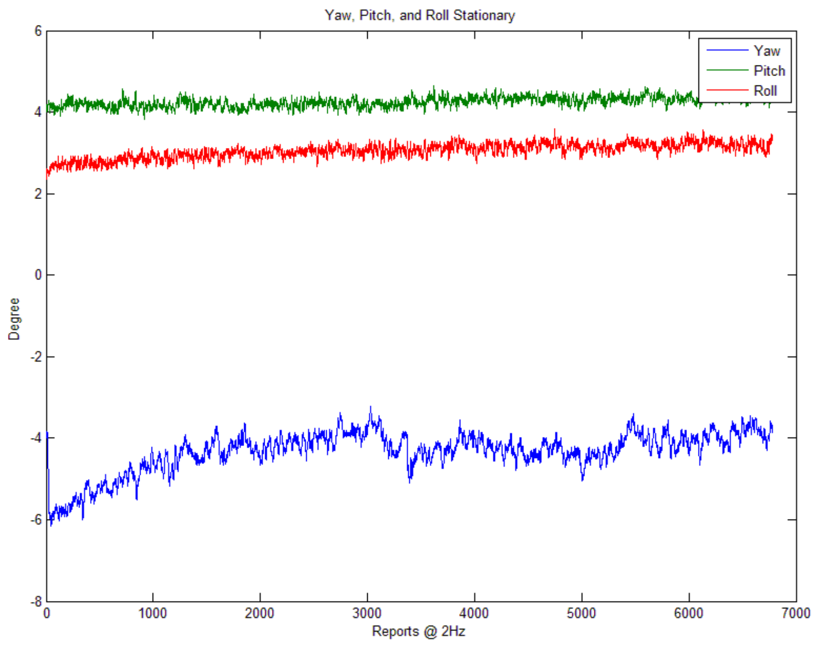


Figure 5.12: Data from a 58 minute stationary test. This plot indicates the amount of drift introduced into the head tracker in an indoor environment with an externally mounted GPS antenna connection.

Unfortunately, due to hardware issues, the body worn computer which was used to run the navigation software and power both the HUD and head tracker, was unable to be used for the testing. Therefore, a Dell laptop was substituted, to replace the body worn computer. Due to this substitution, much more care had to be taken while driving the golf cart and performing the test runs, which greatly affected the timing metrics. In addition, due to the increased processor speed and capabilities of the laptop, the workload on the software did not stress the system enough to indicate any difference in performance. Although all test runs were conducted using the same method, the times varied greatly due to type of terrain, and not the effects of the navigational aide.

One item which became apparent during the test, was the affect of the tunnel in the sky. Due to the path going around obstacles, and having slight variations in altitude due to terrain, the tunnel in the sky was only a slight factor in the traversal of the path. The tunnels did behave as described leading the operator in a straight line from the starting location to the target landing zone. The tunnel appeared to indicate the shortest path, but due to the obstacles on the course, several times the tunnel was not followed. The tunnel did assist in reacquiring the target location. While the target location may have been off the screen, the tunnel acted as a second navigational aide (in addition to the vector arrows), to help the operator reacquire the target location.

5.12 Analysis

The evaluation of the system did uncover issues which affect this research. First, the head tracking software does perform the correct algorithms to translate the physical world locations into the correct screen locations allowing the operator's head movements to be tracked and the screen picture to be correctly updated. The head movements of an operator on the ground relies much more on yaw information, than would a parachutist in 3 dimensional space. A preliminary analysis of the tracking information indicates that all three axis are correctly translated. One issue, which is very evident in the ground testing, is the unreliability of the head tracking information being sent from the MIDG's magnetometer. This is not an insurmountable issue, but needs to have further direct research applied to ensure invalid reports do not propagate.

The testing of the tunnel in the sky as a navigational aid did provide an increase in overall situational awareness, by providing the operator with navigational cues which allows him to keep the target tracking within his field of view and to reacquire the target. This can prove to be extremely useful.

One item was noted during the operational period of the body worn computer. The processor speed was far less than what is appropriate for a system which needs to render head movements in a real-time manner. The screen refresh rate was about 1 Hz, meaning the screen refreshed one time per second, which is far too slow for an operator to accurately and comfortably track locations in a moving situation.

5.13 Conclusion

A portable NVG HUD system has been developed which is designed to provide the operator with a GPS solution to reach a desired landing zone. The solution incorporates both wind and jump parameters to develop a viable navigational path. The system also implements a head tracking system, allowing the operator to look away from the target and have directional indicators which will allow the reacquisition of the target with minimal effort. To build confidence in the system, ground testing was selected as the safest and most appropriate approach for the initial system evaluation. This allows the hardware components and software to be exercised in a safe and controlled environment in order to provide initial capability assessments. In order to build additional confidence in the system, a second step would be to develop a simulation which incorporates all aspects of the algorithm. Once the system has been fully lab tested, the final test would then be to have the entire system jump tested by a qualified individual with appropriate safety considerations in place.

Although the system was tested under limited implementation, the preliminary results indicate the system provides an increased navigational benefit. The head tracking systems did track the target with the proper image translation. A problem does exist with the head tracking hardware, but this can be corrected with some additional programming filters and raw navigational data manipulation. The tunnel in the sky display also proved to be useful in navigating to a target location. The tunnel display provides an additional navigational cue which allows the operator to reacquire the target once it has moved off the visible area. In addition, the vector arrows provide accurate vectors for target location reacquisition, which again, is a significant benefit by letting the operator look away from the target and then be able to reacquire the target.

VI. Conclusions and Recommendations

The goal of this research was to produce a proof of concept for the integration of a GPS display into a paratrooper's night vision goggles heads up display. This would provide an increased capability, allowing operators to reach target landing zones in adverse conditions, such as low light or inclement weather, which would have previously prevented missions.

6.1 Conclusions of Research

The research did accomplish the initial goal of producing a system which could fit within an operator's ammo pouch and produce the navigational message which would allow missions to proceed under low light or inclement weather.

The hardware assembled consisted of a 600 MHz ARM processor module with on an onboard graphics chip. This provided the color display needed to run the AVN-21 HUD. In addition, the module was mated with a third party base board used to provide the external connections such as video, serial, and USB ports. Within the same enclosure the TechGear HUD circuit was also mounted to reduce the number of external cable connections which could foul the operator's parachute. The entire system was powered by a 12 volt lithium polymer battery, which provided enough power to run the system for over 2 hours. Figure 6.1 is the body worn computer in the ammo pouch and Figure 6.2 shows the body worn computer with the cover removed.

The Linux operating system which was developed for this project is based on the Kontron (maker of the processor module) kernel template. The kernel was compiled to run on the ARM processor with minimal additional processes to keep the kernel memory footprint as small as possible. In addition, the navigational application software was developed on an IBM x86 processor and cross-compiled to run on the targeted XScale ARM processor.

The application software is written in C++, utilizing the Eclipse Integrated Development Environment version 3.2. The code was written using object oriented

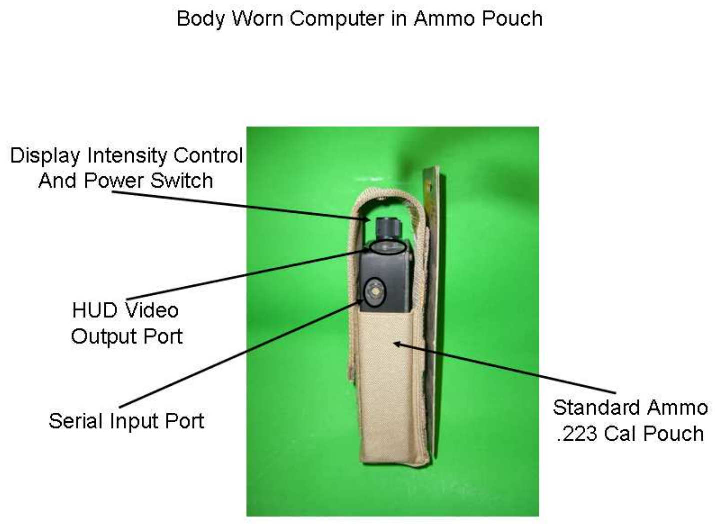

Body Worn Computer in Ammo Pouch
Display Intensity Control And Power Switch
HUD Video Output Port
Serial Input Port
Standard Ammo .223 Cal Pouch

Figure 6.1: Body worn computer in .223 Cal ammo pouch.

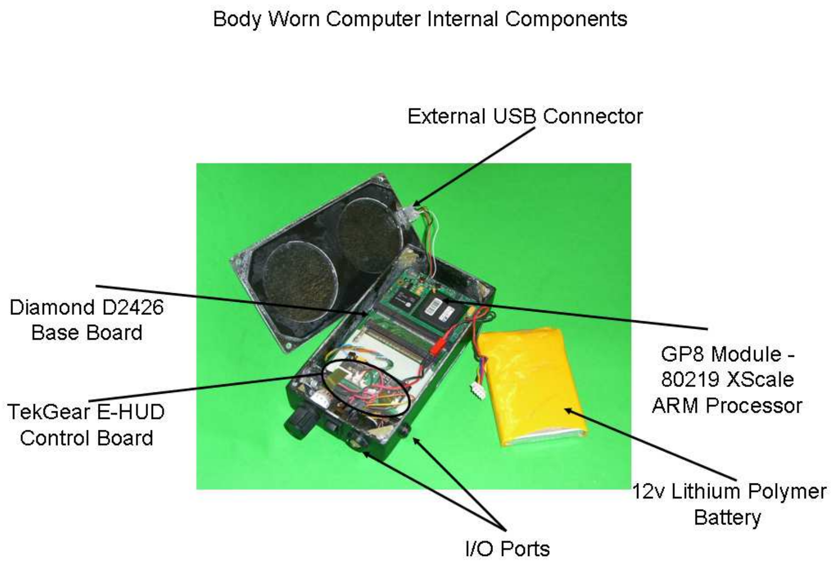

Body Worn Computer Internal Components
External USB Connector
Diamond D2426 Base Board
GP8 Module - 80219 XScale ARM Processor
TekGear E-HUD Control Board
12v Lithium Polymer Battery
I/O Ports

Figure 6.2: Internal components of the body worn computer.

practices, such as interface and model, view, controller patterns. This allowed the software to be developed in an iterative method, testing small modules as the entire system was built. The navigation algorithm takes inputs, such as wind, user parameters, and target locations, to build a navigational solution which is presented to the operator in a tunnel in the sky display.

The system developed provided the operator with a navigational message consisting of a tunnel in the sky leading to a designated target landing zone. The hardware did fit into the required enclosure size and provided enough functional power for the intended time frame. The helmet mounted sensor and antenna provided an accurate estimate of position and orientation. In addition, the software did correctly process the data, which was transmitted from the head tracking system, and produced an accurate target picture. However, there is some issue with the attitude data from the head tracking system.

The second major issue discovered during the field evaluations was the head tracker's disorientation problems. This caused the target to appear 180 degrees off from its correct location. This issue may be caused by a problem with the head tracker's internal magnetometer and how it distinguishes 180 degrees from negative 180 degrees. This issue can be corrected by eliminating the use of the magnetometer and developing a software algorithm to provide orientation based on raw measurements.

The final issue, which arose during the evaluation of the system, was the hardware's lack of stability. During the test, the HUD circuitry failed, which caused the signal from the body worn computer to no longer be able to output a video signal to the night vision goggles. The hardware composition developed during this research provides the necessary component, but has not yet reached its maturity. Several issues arose from attempting to interface components which were manufactured by three different vendors, which caused significant delays in the system's overall completion.

6.2 Significance of Research

This research and evaluation of the developed system did provide a proof of concept that a GPS display could be integrated into a night vision HUD. There are some issues which still need to be addressed and worked out, but the foundation for a self-contained system which provides the user with navigation data is viable. Once fully implemented, this research will give airborne personnel a much broader mission capability by allowing them to proceed in conditions which are currently prohibited. Combat Search and Rescue units will be able to proceed to downed personnel regardless of lighting conditions. Currently, if visibility prevents a clear visual of the landing zone from a given altitude, the mission can not proceed. With this technology, the window of operational missions will be greatly expanded to allow these previously impossible missions. In addition, paratroopers who are making HAHO jumps can proceed in darkness without having to rely on visual ground indicators. This technology will give them a path which accounts for their travel over the distances covered during these long range air movements. The path generated factors items, such as wind and parachute characteristics, to allow the airborne personnel to reach the desired landing zone without utilizing ground cues.

6.3 Recommendations for Future Research

To continue moving this project forward to the eventual live jump test and deployment of the system, several items must first occur. The first step is to develop a simulation which can be used to verify the navigation algorithm. The simulation will have to emulate winds and positional movements while still taking head tracking data from the head tracker. This needs to be accomplished to build confidence and stress the system prior to any real world jump is attempted.

The second item which needs to be addressed is the magnetometer data. Further study of the issues which cause the yaw discrepancy needs to occur. To prevent reliance on the magnetometer data, software needs to be developed in house which produces the inertial measurements. By replacing the magnetometer data with in house developed software, true debugging can occur and confidence in the system reliability can be greatly increased.

The final area for continued research is the processing hardware. In the 18 months since this research began, smaller, more powerful processors have been released. The hardware must be upgraded to to a faster processor, with integrated floating point math. This will allow the update rate to be much faster and smoother. The next version should use an integrated processor chip and mother board to eliminate the interface problems which arise when mating products from different vendors. For example, processors such as the VIA C7 1.GHz processor are currently available which are mounted on a motherboard with all necessary external I/O. An additional benefit is that these processers run standard Windows or Linux operating systems, which removes the need for cross-compilation. This would provide the increased speed while removing unnecessary complexity in software development.

Once these steps have been completed and confidence in the system operation is high, can actual jump testing be considered. Testing the system by a live jump test should only be persuaded once the system has been fully stressed and software bugs have be resolved. Qualified testing agencies such as Natick Soldier Center have specially trained individuals to test these types of systems.

Appendix A. Sample Parameter File

This appendix contains the sample parameter file which is originally read by the system. These parameters are used by the system to set jump specifics.

```
// Config.txt contains user parameters that are set prior to departing aircraft
// Parameters read in at system start up,
// a reboot is needed in order to reload parameters
// All tags must be entered exactly as shown without modification
//
// Parachute safety margin (max / min travel distance percentage);
PARACHUTE_TRAVEL_ERROR=.5
//
// Final approach information
// FINAL_APPROACH_AGAINST_WIND (YES=1 / NO = 0)
// This gives the operator choice of landing against wind
FAP_AGAINST_WIND=1
//
// FINAL_APPROACH_ALTITUDE is the altitude the system stops guidence
FAP_ALTITUDE=400
//
// FINAL_APPROACH_HEADING is the heading to
// approach LZ from Final Approach Point
FAP_HEADING=120
//
// AIM_POINT_HEADING is the heading from Aim
// Point to Final Approach Point
AIM_POINT_HEADING=120
//
// AIM_POINT_ALTITUDE is the altitude delta from
// Aim Point to Final Approach Point
```

```
AIM_POINT_ALTITUDE=1100
//
// AIM_POINT_AGAINST_WIND (YES=1 / NO = 0)
// This gives the operator choice of landing against wind
AIM_POINT_AGAINST_WIND=1
//
// View options
//
// Display Demensions–STS E-Hud SVGA Default 800 x 600
// Display width
SCREEN_WIDTH=800
//
// Display height
SCREEN_HIEGHT=600
//
// BOX_WIDTH is the width of the boxes used
// for the tunnel in the sky
BOX_WIDTH=50
//
// BOX_HEIGHT is the height of the boxes used
// for the tunnel in the sky
BOX_HEIGHT=50
//
// BOX_SEPEARATION is the physical distance
// between the box representations
BOX_SEPERATION=500
//
// END OF CONFIGURATION FILE config.txt
```

This appendix describes the HAHO Navigation System Mission Planner. The mission planner was developed by the Naval Research Center in Pensicola Florida and is currently utilized by the 1st Marine Recon during HAHO missions. The mission planner provides a GUI interface in order to enter wind data.

B.1 HAHO 1D Wind Format

name1,lat1,long1,alt1, name2,lat2,long2,alt2,name3,lat3,long3,alt3,name4,lat4,long4, alt4,ReleasePoint,latRP,longRP,OpenAltitude,DDist,DDir,Jnum, Jtime,Aspd,MOS,latCircle,longCircle,PT,Wmag0,Wmag1K,,Wmag2K, Wmag3K,Wmag4K,Wmag5K,Wmag6K,,Wmag7K,Wmag8K,Wmag9K,Wmag10K, Wmag12K,Wmag14K,Wmag16K,Wmag18K,Wmag20K,Wmag22K,Wmag24K, Wmag26K,Wmag28K,Wmag30K,Wdir0,Wdir1K,,Wdir2K,Wdir3K,Wdir4K, Wdir5K,Wdir6K,Wdir7K,Wdir8K,Wdir9K,Wdir10K,Wdir12K,Wdir14K, Wdir16K,Wdir18K,Wdir20K,Wdir22K,Wdir24K,Wdir26K,Wdir28K,Wdir30K, Pname,ParSpeed,PFallRate,cbPar.SelectedIndex,SpeedUnits,DistanceUnits, AltitudeUnits,Log,Jump,JMode,filepath,

B.2 Sample 1D Wind File

Prime Landing Zone,39.88240166,-84.1809016,120,NOWAYPOINT,0,0,0, NOWAYPOINT,0,0,0,NOWAYPOINT,0,0,0,ReleasePoint,39.88243333, -84.182283333,1944,8,12,10,5,150,10,34.0555570128726,69.9893790092627, 9.375,299999,30,31,32,33,34,30,36,37,38,39,40,41,42,43,44,45,46,47,48,49, 0,45,90,180,270,360,405,720,722,719,60,61,62,63,64,65,66,67,68,69,70, mc,25,1200,1,1,1,1,0,1,2,C: Documents and Settings Administrator My Documents MP files 9.txt

Table B.1: HAHO Navigation System Mission Planning File Format.

Location	Variable Name	Description
0	name1	Name of the first LZ
1	Lat1	Latitude of the first LZ in D.dd, N=+, S=-
2	long1	longitude of the first LZ in D.dd, W=+, E=-
4	alt1	altitude of the first LZ in ft MSL
16	Release Point	Name of the Release Point
17	latRP	latitude of the Release Point in D.dd, N=+,S=-
18	longRP	longitude of the Release Point in D.dd, W=+,E=-
19	OpenAltitude	Opening altitude of the Release Point in ft MSL
20	DDist	Distance from RP to Prime LZ in nm
21	DDir	Direction from RP to Prime LZ in degrees
28	PT	Parachute Travel Distance in NM
29	Wmag0	Wind Magnitude at 0 ft MSL
30	Wmag1K	Wind Magnitude at 1K ft MSL
31	Wmag2K	Wind Magnitude at 2K ft MSL
32	Wmag3K	Wind Magnitude at 3K ft MSL
33	Wmag4K	Wind Magnitude at 4K ft MSL
34	Wmag5K	Wind Magnitude at 5K ft MSL
35	Wmag6K	Wind Magnitude at 6K ft MSL
36	Wmag7K	Wind Magnitude at 7K ft MSL
37	Wmag8K	Wind Magnitude at 8K ft MSL
38	Wmag9K	Wind Magnitude at 9K ft MSL
39	Wmag10K	Wind Magnitude at 10K ft MSL
40	Wmag12K	Wind Magnitude at 12K ft MSL
41	Wmag14K	Wind Magnitude at 14K ft MSL
42	Wmag16K	Wind Magnitude at 16K ft MSL
43	Wmag18K	Wind Magnitude at 18K ft MSL
44	Wmag20K	Wind Magnitude at 20K ft MSL
45	Wmag22K	Wind Magnitude at 22K ft MSL
46	Wmag24K	Wind Magnitude at 24K ft MSL
47	Wmag26K	Wind Magnitude at 26K ft MSL
48	Wmag28K	Wind Magnitude at 28K ft MSL
49	Wmag30K	Wind Magnitude at 30K ft MSL
50	Wdir0	Wind Direction (from) at 0 ft MSL
.	.	.
.	.	.
.	.	.
70	Wdir30K	Wind Direction (from) at 30K ft MSL
71	Pname	Name of the Parachute
72	ParSpeed	Speed of the Parachute in knots
73	PFallRate	Fall Rate of the Parachute in ft/min

Bibliography

AND00. RIVER ANDREAS. Gnugo sdl graphics library. Jan 2000. SGE Graphics for rectangles and fonts.

ANS07. ANSWERS. Head-up display. June 2007. A basic history of HUD systems and their implementation.

ARM05. ARM. Arm procuct backgrounder. January 2005. An overview of the ARM architecture.

BAL03. BRIAN BALAZ. A three-dimensional heads-up primary navigation reference display for paratroopers performing hight altitude high open jumps. Master's thesis, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, May 2003. AFIT/GE/ENG/03.

BOL03. DOUGLAS BOLING. *Programming Microsoft-Windows CE.NET*. Number 0-7356-1884-4. Microsoft Press, One Microsoft Way, Redmond, Washington 98052-6399, third edition edition, 2003.

CHE07. GRIBBLE CHERYL. A brief history of c. Jan 2007. A history of the C programming languages.

CRA08. JASON CRALEY. Air drop systems. Technical report, Natick Soldier Research, Development and Evlauation Center, Natick, MA, 2008.

DAN98. PETER DANE. Eeng 533: Navigation using the gps course handsout spring 2006. Technical report, Air Force Institute of Technology, Wright Patterson AFB, OH, 1998.

DAN08. REGINALD DANIELS. Head tracking small business innovative resources, May 2008. This was the initial SBIR meeting for the Phase I head tracking and noise modeling technology.

DAV07. BOB DAVIS. Night vision. *American Cop*, 3(2):38–41, March/April 2007.

DRA06. LABORATORY DRAPER. *Joint Precision Airdrop System Mission Planner*. Number 423288. 555 Technology Square, Cambridge, Massachusetts 02139, revision 10.0 edition, 2006.

ELE08. GLENN ELERT. The nature of light. Jan 2008. Explanation of the Monochromatic light.

FER04. ANDREW FERGUSON. The history of computer programming languages. Nov 2004. A history of programming languages from Babage to current high level language.

FLE07. JORDAN FLETCHER. Real-time gps-alternative navigation using commodity hardware. Master's thesis, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, May 2007. AFIT/GE/ENG/02.

GEN08. HAHO GENTEX. Gentex haho helmet home page. Jan 2008. Description of Gentex HAHO helmet.

HWA92. YONG HWANG. Gross motion planning–a survey. Technical report, Sandia National Labaratories, Albuquerque, New Mexico, 1992.

INT06. INTEL. Intel 80219 general purpose pci processor. January 2006. 80219 Processor specs.

INT07a. INTEL. Intel xscale microarchitecture technical summary. January 2007. XScale Microarchitecture specs.

INT07b. DIAMOND POINT INTERNATIONAL. D2426 x-board baseboard. Aug 2007. Diamond Point International D2426 homepage.

KAP06. HEGARTY KAPLAN. *Understanding GPS Principles and Applications*. Number 1-58053-894-0. Artech House, Artech House, Inc, 685 Canton Street, Norwood, MA 02062, second edition edition, 2006.

KON06. KONTRON. Resource guide 2006. January 2006. PC 104 and Small Form Factors.

KRA98. LYNDA KRAMER. Pathway design effects on synthetic vision head-up displays. Research paper, Nasa Langley Research Center, 24 West Taylor Street, Hampton, VA, USA, 1998.

LIN08. TIMESYSTEM LINUX. Introduction to cross-compiling for linux. Jan 2008. A history of cross compiling.

MET03. STEVEN METSKER. *Design Patterns in Java*. Number 0-7356-1884-4. Addison-Wesley, One Microsoft Way, Redmond, Washington 98052-6399, third edition edition, 2003.

MIC07. MICROBOTICS. Midg ii homepage. June 2007. The MIDG II home page.

MIS06. ENGE MISRA. *Global Positioning System Signals, Measurements, and Performance*. Number 0-9709544-1-7. Ganga-Jamuna Press, Ganga-Jamuna Press P.O. Box 692 Lincoln, Massachusetts 01773, second edition edition, 2006.

PAR03. RUSSEL V PARRISH. Avionic pictorial tunnel pathway highway in the sky workshop. Jan 2003. Four hud workshop.

PRO05. LINUX INFORMATION PROJECT. Kernel defination. May 2005. The Kernel defination.

PRO07. LINUX INFORMATION PROJECT. About linus torvalds. Sept 2007. The Linus Torvalds bio.

QSA08. UK QIOPTIQ St Aspha. Qioptiq product homepage. Jan 2008. Maker of HUD military products homepage.

SEC06. GLOBAL.ORG SECURITY. Parachute history. March 2006. A history of the parachute from the time of da Vinci to Ram air systems.

SEC07. GLOBAL.ORG SECURITY. Military free fall school. Aug 2007. John F. Kennedy Special Warfare Center- Military Free Fall School.

SNO99. MICHAEL SNOW. Flying complex approaches using a head-up display: Effects of visibility and display type. Research paper, United States Air Force Research Laboratory, 2255 H Street,Crew System Interface Division, Wright-Patterson AFB, OH 45433-7022, 1999.

SPI07. CARY SPITZER. *Digital Avionics Handbook, 2d ed.* Number 0-849384419. CRC Press, One Microsoft Way, Redmond, Washington 98052-6399, second edition edition, 2007.

STS07. STS. Av/pvs21 homepage. June 2007. The AV/PVS21 home page.

TEK06. TEKGEAR. Svga-3d oled microdisplays. Jan 2006. Technical specifications for the EHUD utilized by the STS company for the HUD.

THO05. JASON THOMPSON. A three-dimensional helmet mounted primary flight reference for paratroopers. Master's thesis, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, May 2005. AFIT/GE/ENG/05.

UNI07. PORTLAND STATE UNIVERISTY. Ins home-portland state university. Aug 2007. The PSAS Inertal Navigation System home page.

VET. MICHAEL VETH. Master's thesis, Wright-Patterson AFB OH, September, note = AFIT/EENG/02, school = Graduate School of Engineering, Air Force Institute of Technology (AETC), title = Fusion of Imaging and Inertial Sensors for Navigation, year = 2006,.

Vita

Captain Fernando Ontiveros attended Park College where he graduated with honors and received a Bachelor of Science degree in Computer Science in May 2000. Following graduation and completion of Officer Training School he was commissioned as a Second Lieutenant in the United States Air Force.

Captain Ontiveros served for two years at Vandenburg AFB, CA as a Squadron Section Commander. In August 2002, he was assigned to US Forces NATO at Supreme Headquarters Allied Powers Europe (SHAPE), where he deployed to Afghanistan and Iraq. In August 2006, Captain Ontiveros entered the Air Force Institute of Technology (AFIT) as a graduate student in the Department of Electrical and Computer Engineering. He graduated from AFIT with a Masters of Science degree in Computer Science with a focus on Software Engineering.

Permanent address: 2950 Hobson Way
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433

Index

The index is conceptual and does not designate every occurrence of a keyword. Page numbers in bold represent concept definition or introduction.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 19–06–2008	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* September 2006 — June 2008

4. TITLE AND SUBTITLE Development of a Night Vision Goggle Heads Up Display For Paratrooper Guidance	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Fernando Ontiveros, Captain, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way, WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GCS/ENG/08-24
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Mr. Robert Stephens, Sensor Technology Systems, 2794 Indian Ripple Road, Beavercreek, Ohio 45440, USA, (937) 426-2341 e-mail: robert.stephens@ogaragroup.com	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis provides the proof of concept for the development and implementation of a Global Positioning System (GPS) display via Night Vision Goggles (NVG) Heads-Up Display (HUD) for paratroopers. The system has been designed for soldiers who will be able to utilize the technology in the form of a processing system worn in an ammo pouch and displayed via NVG HUD as a tunnel in the sky. The tunnel in the sky display design is essentially a series of boxes displayed within the goggle's HUD leading the paratrooper to the desired Landing Zone (LZ). The algorithm developed is effective and efficient in order to receive GPS sensor data, correlate head-tracking data, and display the combined information in the paratrooper's NVG HUD as the tunnel in the sky. The primary goal of the project is to provide a product which allows Special Operations personnel to reach a desired LZ in obscured visibility conditions, i.e. darkness, clouds, smoke, and other unforeseen situations. This allows missions to be carried out around the clock, even in adverse visibility conditions which would normally halt operations.

15. SUBJECT TERMS

GPS, Paratroopers, Heads Up Display, HUD, Global Positioning System, Night Vision Goggles, NVG,

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			John F. Raquet, Ph.D. (ENG)
U	U	U	UU	140	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255–3636, x4580; john.raquet@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18