



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

GEOMETRY-OF-FIRE TRACKING ALGORITHM FOR DIRECT-FIRE WEAPON SYSTEMS

by

Caleb K. Khan

September 2015

Thesis Advisor:

Co-Advisor:

Second Reader:

Zachary Staples

Xiaoping Yun

James Calusdian

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 2015	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE GEOMETRY-OF-FIRE TRACKING ALGORITHM FOR DIRECT-FIRE WEAPON SYSTEMS			5. FUNDING NUMBERS	
6. AUTHOR(S) Khan, Caleb K.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) The continuous cycle for validating geometry of fires on a battlefield impedes momentum. A force that can decide and act quicker than the enemy has an advantage. The goal of this thesis is to provide dismounted infantry units with this advantage by developing a sensor network that streamlines the geometry-of-fire validation cycle for direct-fire weapon systems. This is the first attempt to develop technology for this specific application. Prototyping the system's overall design became this thesis' main objective. A kinematic model of a rifleman was created to describe the motion of a manipulated weapon system, and a geometry-of-fire tracking algorithm was created to compute the offsets between friendly nodes in a network. The kinematic model and tracking algorithm were both verified using YEI 3-Space Data-Logging sensors and the VICON motion system. The sensor, model, and algorithm were integrated into a tactical network that was designed by concurrent research at the Naval Postgraduate School. Assuming an inertial personal navigation system can be created, we found that this basic prototype provided a viable foundation for further development. The results of this thesis demonstrate the potential for this technology to be fully developed and implemented to enhance dismounted infantry units.				
14. SUBJECT TERMS Geometry of fire, situational awareness, close quarters combat, direct-fire weapon system, tactical network, personal navigation system, orientation, kinematics, coordinate system, reference frame, transformation operator, quaternion, rotation matrix, Euler angles, YEI, VICON.			15. NUMBER OF PAGES 107	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

**GEOMETRY-OF-FIRE TRACKING ALGORITHM FOR DIRECT-FIRE
WEAPON SYSTEMS**

Caleb K. Khan
Captain, United States Marine Corps
B.S., United States Naval Academy, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

**NAVAL POSTGRADUATE SCHOOL
September 2015**

Author: Caleb K. Khan

Approved by: Zachary Staples
Thesis Advisor

Xiaoping Yun
Co-Advisor

James Calusdian
Second Reader

R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The continuous cycle for validating geometry of fires on a battlefield impedes momentum. A force that can decide and act quicker than the enemy has an advantage. The goal of this thesis is to provide dismounted infantry units with this advantage by developing a sensor network that streamlines the geometry-of-fire validation cycle for direct-fire weapon systems. This is the first attempt to develop technology for this specific application. Prototyping the system's overall design became this thesis' main objective. A kinematic model of a rifleman was created to describe the motion of a manipulated weapon system, and a geometry-of-fire tracking algorithm was created to compute the offsets between friendly nodes in a network. The kinematic model and tracking algorithm were both verified using YEI 3-Space Data-Logging sensors and the VICON motion system. The sensor, model, and algorithm were integrated into a tactical network that was designed by concurrent research at the Naval Postgraduate School. Assuming an inertial personal navigation system can be created, we found that this basic prototype provided a viable foundation for further development. The results of this thesis demonstrate the potential for this technology to be fully developed and implemented to enhance dismounted infantry units.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	TECHNOLOGICAL SUPERIORITY IN WAR	1
B.	GEOMETRY OF FIRE.....	1
C.	SENSOR NETWORK	3
D.	THESIS OBJECTIVE	5
II.	BACKGROUND	7
A.	ASSUMPTIONS.....	7
1.	Personal Navigation System.....	7
2.	Tactical Network.....	11
3.	CQC Considerations.....	12
B.	CONE OF FIRE	13
C.	ALGORITHMS	17
1.	Reference Coordinate System.....	17
2.	Model of a Rifleman’s Kinematics	18
3.	Geometry-of-Fire Tracking Algorithm.....	24
D.	SUMMARY	28
III.	EXPERIMENTAL DESIGN.....	31
A.	HARDWARE COMPONENTS.....	31
1.	YEI 3-Space Data-Logging Sensor	31
2.	VICON Motion System.....	32
B.	TEST PLAN	33
1.	Validation of the Sensor’s Accuracy	34
2.	Verification of the Rifleman’s Kinematic Model	37
3.	Testing of the Geometry-of-Fire Tracking Algorithm	38
C.	SUMMARY	40
IV.	RESULTS AND ANALYSIS	41
A.	VALIDATION OF THE SENSOR’S ACCURACY	41
1.	Synchronizing the Collected Data Sets	41
2.	Accuracy of the Sensor’s Estimated Orientations	44
B.	VERIFICATION OF THE KINEMATIC MODEL	47
1.	Scaling Factor for the Kinematic Model.....	47
2.	Trueness of the Kinematic Model.....	49
C.	TESTING OF THE TRACKING ALGORITHM.....	51
1.	Simulation of Moving Friendly Nodes	51
2.	Algorithm’s Ability to Detect Geometry Issues.....	52
D.	SUMMARY	53
V.	CONCLUSION AND RECOMMENDATIONS.....	55
A.	SUMMARY OF CONTRIBUTIONS.....	55
B.	FUTURE WORK.....	56
	APPENDIX.....	59

A.	SETTINGS FOR THE YEI 3-SPACE DATA-LOGGING SENSOR.....	59
B.	MATLAB SCRIPTS FOR VERIFYING THE KINEMATIC MODEL ..	61
1.	Function for Analyzing the Sensor’s Accuracy.....	61
2.	Function for Synchronizing the Collected Data Sets	65
3.	Main Script File for Implementing the Kinematic Model	66
C.	PYTHON SCRIPTS FOR TESTING THE TRACKING ALGORITHM.....	77
1.	Function for Publishing Node Positions.....	77
2.	Script File for Implementing Tracking Algorithm	78
	LIST OF REFERENCES	87
	INITIAL DISTRIBUTION LIST	89

LIST OF FIGURES

Figure 1.	The prototype system design of the sensor network includes five components. After [1].	4
Figure 2.	Radio-based ranging and cooperative navigation demonstrate potential for solving the indoor positioning problem. From [4].	9
Figure 3.	Two units use cooperative navigation to refine their positions as they walk around a closed-loop corridor. From [4].	10
Figure 4.	A block diagram that depicts the data flow between the network's core and the riflemen nodes. After [5].	11
Figure 5.	The trajectory pattern of a fired round is described by a cone of fire. From [7].	13
Figure 6.	A rifleman implements the <i>300 Mil Rule</i> by placing a thumb on the front-sight post and extending the pinky finger. After [8], [9].	15
Figure 7.	Two-dimensional representation of the vectors and angles that are used to determine if a geometry-of-fire issue exists.	16
Figure 8.	An M4 carbine mounted to a tripod is used to model a rifleman's kinematics. After [12]–[15].	19
Figure 9.	Four coordinate systems are attached to the weapon system and tripod for the kinematic model. After [12]–[15].	21
Figure 10.	Translation vectors between the four coordinate systems that are attached to the weapon system and tripod. After [12]–[15].	22
Figure 11.	The main steps of the geometry-of-fire tracking algorithm.	25
Figure 12.	The 3-Space Data-Logging sensor from YEI Technology tracks the weapon systems' orientations for the network. From [15].	32
Figure 13.	Ground-truth data for the experiments is provided by the VICON motion system. From [18].	33
Figure 14.	Sufficient standoff is maintained at both ranges despite the sensor's maximum error. After [8].	35
Figure 15.	The weapon system's visible laser was held at these various grid coordinates in order to validate the sensor's accuracy for estimating static orientations. After [8].	36
Figure 16.	The weapon system's visible laser was moved along the perimeter of this non-symmetrical polygon in order to validate the sensor's accuracy for estimating dynamic orientations. After [8].	37
Figure 17.	The geometry-of-fire tracking algorithm was tested for these four situations. After [12]–[15].	39
Figure 18.	The collected data sets from the sensor and VICON motion system must be synchronized in order to validate the sensor's accuracy.	42
Figure 19.	Synchronizing the data sets from the sensor and VICON motion system provided a one-to-one comparison of the orientations that are described by their Euler angles.	43
Figure 20.	An angular error of $\pm 5^\circ$ causes insufficient standoff at both ranges. After [8].	45

Figure 21.	The sensor's error reduces as the weapon system's orientation is fixed or its rate of change is slowed.	46
Figure 22.	A scaling factor adjusted the length of the laser's vector to ensure that its endpoint remained in the two-dimensional Cartesian plane as the weapon system was manipulated. After [12]–[15].	48
Figure 23.	The model's description of the weapon system's orientation as its PEQ-15 visible laser traces the polygon's perimeter.	50
Figure 24.	An example of the <i>friendly detection</i> script's output that alerts a rifleman of a present geometry issue.	52
Figure 25.	The default coordinate system was adjusted to align its coordinate directions with the NED reference frame as depicted. From [21].	59
Figure 26.	The sensor's filter mode, running average, and reference mode were adjusted as shown. After [21].	60

LIST OF TABLES

Table 1.	The sensor's accuracy for estimating orientations under static and dynamic conditions.....	44
----------	--	----

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ACOG	Advanced Combat Optical Gunsight
CQC	close quarters combat
GPS	Global Positioning System
IMU	inertial measurement unit
NED	north-east-down
NPS	Naval Postgraduate School
NVG	night vision goggles
PEQ-15	AN/PEQ-15 Advanced Target Pointer/Illuminator
PNS	personal navigation system
ROS	Robot Operating System
UWB	ultra wideband

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Accomplishing this thesis was an extremely rewarding experience for me. As a student at the Naval Postgraduate School, I was presented with a unique opportunity to merge my military experience with the faculty's academic knowledge. This enabled me to explore technology that may provide a tactical advantage to dismounted infantry units. Having the chance to impact the individual rifleman meant a lot to me as a Marine infantry officer. This would not have been possible without the support and direction of my advisors and second reader. My advisor, Zachary Staples, turned this novel idea into a viable research topic and guided me through the design of the overall system. The expertise of my co-advisor, Xiaoping Yun, helped me overcome challenges with developing the kinematic model and tracking algorithm. My second reader, James Calusdian, offered me daily mentorship and granted me full access to his laboratory facility and equipment for this research. I am thankful for all of their time and effort throughout this process.

I would also like to thank my family for their endless encouragement and sharing this experience with me.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. TECHNOLOGICAL SUPERIORITY IN WAR

A technological advantage in war serves as a force multiplier. The side that maintains technological superiority is not guaranteed victory, but their forces will be enhanced and have the capability to be more effective. When all other things are equal, the side with this technological superiority will have an edge, and sometimes this edge is all that is needed to tip the scales of war in their favor. In World War I, airplanes bettered reconnaissance capabilities, and tanks brought shock and mobility to the battlefield. In the Korean War, helicopters added flexibility for logistical resupplies, medical evacuations, and command and control nodes. These platforms are examples of newly developed technologies that provided an edge in war. The purpose of this thesis research is to develop a sensor network that provides a similar technological advantage to the individual rifleman.

Dismounted infantry benefit from technological advancements that enable them to be more effective. The Global Positioning System (GPS), night vision goggles (NVG), and various rifle optics are notable examples. GPS streamlined land navigation for dismounted infantry units. NVGs provided more stealth and surprise by enabling military operations to expand into poorly illuminated conditions. Rifle optics led to improved marksmanship, target illumination, and an accurate means to acquire targets that cannot be observed directly or engaged with direct-fire weapon systems. This thesis' sensor network will add to these examples by mitigating geometry-of-fire issues for riflemen's direct-fire weapon systems. Providing dismounted infantry units with a means to track their geometry of fires will dramatically enhance their situational awareness.

B. GEOMETRY OF FIRE

Geometry of fire is a constant concern during combat engagements. It is a military term that refers to the physical offset between the orientations and positions of friendly units on a battlefield. This offset must be properly maintained at all times in order to mitigate the potential for fratricide. This concern is engrained into Marine Corps officers

at the Basic Officer Course and Marine riflemen in the Fleet Marine Force. It incorporates the concept of coordinate systems. Units must be able to describe their orientations and positions with respect to a common reference frame and ensure that their weapons' effects do not interfere with each other. It is a scalable concept that applies to an individual as much as it does to a unit. Riflemen must be aware of their individual geometries with respect to their fellow soldiers just like unit leaders must be aware of their unit's geometries with respect to their adjacent units.

Geometry of fire is a consideration that affects every phase of an operation. Unit leaders attempt to mitigate issues with their units' geometries during the planning phase by war-gaming their orientations and positions. They typically step through their planned scheme of maneuver with visual aids to ensure that their geometries are properly maintained while their mission objectives are being met. For example, a unit leader will adjust the unit's plan to flank an objective from west-to-east if he or she knows that an adjacent unit will establish a support-by-fire position to the east within the range of their weapons' effects. In the conduct and exploitation phases, it is everyone's responsibility to continuously validate his or her geometries. Individual riflemen must confirm that their weapons' effects will not endanger their fellow soldiers before engaging enemy targets, and unit leaders must confirm their unit's orientation and position before transitioning to the next part of their scheme of maneuver.

This continuous validation cycle slows a unit's tempo. A tactical pause is typically required if a geometry-of-fire issue exists. An individual rifleman or unit leader must deliberately slow his or her progress in order to visually confirm that they are not going to endanger another friendly force. For individual riflemen, this pause can be fluid as they may just need to visually confirm that their fellow soldiers are safely outside their weapons' effects. For unit leaders, this pause can cause their entire unit to hold its advance as a hasty map study may be required to ensure that their unit is maintaining its proper offsets. Regardless, these pauses take away time and focus from friendly units on the battlefield. This cycle becomes especially challenging and tedious when a unit encounters friction or when the battlespace is cluttered. In an urban environment or aboard a ship, individual rifleman may not be able to see and communicate directly with

everyone else in their unit. This increases the potential for a geometry issue and slows the unit's momentum even more.

Military operations in an urban environment and aboard a ship are the most daunting with respect to geometry of fires. These environments require specialized tactics that are generally known as close quarters combat (CQC). CQC is a military and law enforcement term that refers to small-unit engagements that take place within short ranges. This type of combat requires riflemen to be decisive, quick and precise with the employment of their weapon systems. The confined spaces in these environments make it extremely difficult for friendly units to maintain proper geometry of fires. Clearing interior rooms of a building or a ship can disorient the most experienced riflemen. Without direct lines-of-sight to their fellow soldiers or windows to confirm their offsets with respect to a reference frame, it becomes especially challenging to continuously validate their geometries. A worst-case example is that a rifleman can be oriented at a wall with a friendly unit stacked on the other side of it. The potential for this unnerving scenario can cause hesitation in an environment that demands quick reactions and decision-making due to its close proximities.

C. SENSOR NETWORK

The goal of this thesis research is to streamline the geometry-of-fire validation cycle. Arming individual riflemen with technology that can do this will provide them with an edge during combat. This technology must continuously track every rifleman on the battlefield without any actions required by the actual riflemen. It must be able to identify when geometry of fires are conflicted and to inform riflemen when they fail to maintain proper offsets. Riflemen must retain the ability to fire their weapon regardless of their geometries. This technology will be designed to improve their situational awareness and to help riflemen make informed and quick decisions to employ their weapon systems. It will not serve as a substitute for individual proficiency or training. It should be considered an added capability similar to the way GPS augments riflemen's land navigation skills. This system should give riflemen confidence and reduce hesitation. Since CQC in an urban environment and aboard a ship demands the most confidence and

least hesitation, this type of combat will be explored to provide a challenging framework for this research. It will serve as the standard and set any limiting factors as this system is developed. The assumption is that this technology can be easily adapted to any type of combat or environment if it works for CQC.

The prototype system design for this sensor network is depicted in Figure 1. Every rifleman will serve as a friendly node. The network will store and publish their three-dimensional positions with respect to a common reference frame, and each node will be responsible for continuously tracking their orientation and position, updating their position on the network and subscribing to the network's *friendly positions* message. This configuration requires each node to internally compute its relative offsets and to detect potential geometry issues. Some type of feedback mechanism is required to inform the riflemen when they are failing to maintain proper geometries. For example, this feedback mechanism can be a light turning on in the optic of the rifleman who is oriented at another friendly node. As shown, this design requires five hardware components for each node: a transmitter/receiver to communicate with the network, a sensor to track position, a sensor to track orientation, a micro-processor/controller to compute the offsets, and a feedback mechanism to inform the rifleman.

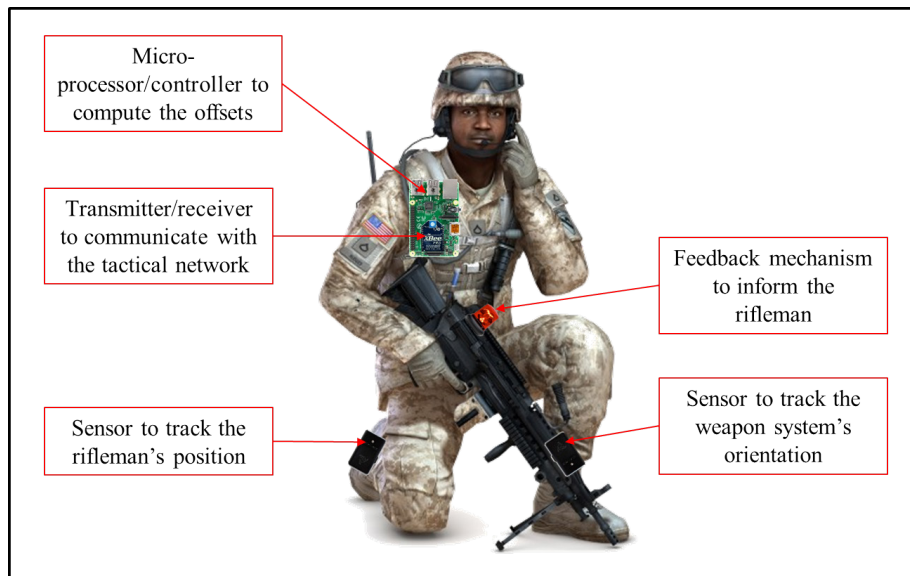


Figure 1. The prototype system design of the sensor network includes five components. After [1].

Prototyping this design is the main objective of the thesis. Researching at the Naval Postgraduate School (NPS) provides a unique opportunity to merge the military experience of its students with the academic knowledge of its faculty. The integration of these five components for this specific application is an original idea that was developed by the students and faculty of the Department of Electrical and Computer Engineering at NPS. This is the first attempt to develop a technology that may streamline the geometry-of-fire validation cycle. Evaluating the overall design of this system is necessary before refining any specific component. Surveying ideal hardware components and analyzing alternate system designs are supporting efforts to be considered in future work. This prototyping process is separated into five topics that correspond to the system's five components. Two are focused upon in this thesis: tracking the weapon systems' orientations and internally computing the offsets at each node. All of these topics will be integrated into a single prototype of the complete system once they have been individually researched and validated.

D. THESIS OBJECTIVE

The continuous cycle for validating geometry of fires on a battlefield impedes momentum. A force that can decide and act quicker than their enemy has an advantage, and the goal of this research is to develop technology that enables this tempo. A network that can track the positions of individual riflemen and inform them when they are oriented at friendly units is designed. It provides riflemen with an edge by helping them to make informed and quick decisions to employ their weapon systems. CQC in an urban environment and aboard a ship demands the most confidence and least amount of hesitation regarding weapon's employment; therefore, CQC serves as a reference frame and establishes limiting factors for the system design. Rapidly prototyping the design is the main objective for this research. This is the first attempt to develop this technology, and validating the overall concept is paramount.

In this thesis, tracking weapon systems' orientations and computing the offsets between each node in the network are specifically focused upon. Assume that the network is properly configured to track and communicate riflemen's positions for a CQC

operation. The validity of this assumption, the challenges of estimating orientations and computing offsets, and the algorithms implemented to accomplish these tasks are discussed in the next chapter. A model of a rifleman's kinematics and a geometry-of-fire tracking algorithm were developed to accomplish the objectives of this thesis. The experimental designs that evaluated this model and algorithm are presented in Chapter III, where the hardware components that were selected for these experiments and the test plans that verified the kinematic model and tracking algorithm are detailed. As the experiments were conducted, challenges were encountered that adjusted the original test plans. These major challenges and the results from each experiment are summarized in Chapter IV. These results are concluded in the final chapter, where the original contribution of this research is emphasized and recommendations for future work are presented.

II. BACKGROUND

The purpose of this chapter is to review the preliminary work that was required to conduct the experiments for this thesis. The assumptions that were introduced are examined in order to show why they are valid suppositions. The problems of tracking weapon systems' orientations and computing offsets between friendly nodes in a network are framed. Finally, the concepts and algorithms used to model a rifleman's kinematics in order to detect when riflemen fail to maintain proper geometries are detailed. A common understanding of this background information is required before introducing the experimental designs used to verify that these orientations and offsets can be estimated and computed.

A. ASSUMPTIONS

Assumptions must be made in order to continue this research. According to the Marine Corps Planning Process, assumptions are only valid if they are logical, realistic and essential to continue. The risk of accepting assumptions must be universally understood, and they must be tracked until they are either validated or disproved [2]. In order to show that they are reasonable suppositions, three assumptions that were introduced in the previous chapter are examined in this section. First, the feasibility of developing a reliable personal navigation system (PNS) that can function in GPS-denied environments for the typical duration of a CQC operation is examined. Second, the feasibility of configuring a network to track and communicate riflemen's positions in an urban environment and aboard a ship is analyzed. Third, the CQC considerations that impact the system's overall design are detailed. Full disclosure of these assumptions and considerations is required in order to properly frame the orientation and offset problems.

1. Personal Navigation System

A PNS must be developed for the network to be able to track riflemen's positions. Assuming that a PNS can be created for this specific application is reasonable and necessary to continue this research. The actual positions of riflemen on a battlefield must be known before attempting to track the orientations of their weapon systems and to

compute the offsets between their geometries. At first glance, this assumption may seem straightforward to develop. Its design could be implemented with GPS like many other military systems. Unfortunately, this PNS is not able to rely solely on GPS due to its restrictive operating environments. GPS requires at least four unobstructed satellite signals to determine fixed positions with enough precision to support dismounted infantry operations [3]. In an urban environment and aboard a ship, receiving four signals may not be possible. These signals quickly attenuate as they propagate through the walls of buildings and bulkheads of ships [3]. Tracking indoor positions is extremely challenging, and this PNS must be reliable, especially in GPS-denied environments.

The feasibility of developing this PNS must be examined further in order to show that it is a valid assumption. Constructing a network to communicate riflemen's positions is pointless if a PNS cannot track their positions inside of buildings and ships. The technology for this specific application may not currently exist, but the potential for it to be developed must be shown. Positioning technologies and techniques were surveyed in order to demonstrate the potential for solving this indoor problem. Foot-mounted inertial measurement units (IMU's) can attain meter-level accuracy in GPS-denied environments with a technique known as zero-velocity updating. This technique corrects the inherent bias and drift from the sensor's accelerometer data when the foot stands still [3]. Unfortunately, this level of accuracy can only be maintained for a short duration, and additional sensors or techniques must be incorporated in order to preserve this accuracy throughout extended CQC operations. Many of the devices that were researched could help. Imaging sensors, barometers, and Doppler radars are some examples of these suitable devices [4]. The overall system must be as lightweight and inexpensive as possible, and it must not rely on a priori map information [4]. Detailed layouts of buildings and ships are rarely known prior to an operation. The system cannot rely on devices that require these types of details or pre-installed infrastructure. Two techniques that show the most promise and meet these constraints are radio-based ranging and cooperative navigation [4].

Radio-based ranging relies on portable infrastructure as shown in Figure 2. In the figure, radios are brought to the scene of an operation by military vehicles [4]. Supporting

elements to the unit conducting the assault are responsible for staging and monitoring these radios. These elements can remain in fixed locations outside of the buildings and ships, or they can follow the assaulting unit's movement and drop the radios within the buildings and ships. Ranges to individual riflemen can then be estimated opportunistically to the radios' transceivers [4]. Ultra wideband (UWB) radios produce accurate estimates, but their range is limited due to the attenuation of signals as they propagate through walls and bulkheads [4]. Cooperative navigation helps with this technique's limited range.



Figure 2. Radio-based ranging and cooperative navigation demonstrate potential for solving the indoor positioning problem. From [4].

Cooperative navigation exchanges positioning estimates and uncertainties between riflemen. Similar to radio-based ranging, this filtering technique opportunistically incorporates estimates and uncertainties whenever they are available [4]. This improves and extends positioning accuracy in a decentralized manner [4]. In Figure 3, the positions of two individuals are tracked as they walk indoors along a closed-loop path. These individuals are equipped with foot-mounted IMU's and UWB radios [4].

As shown, their measured trajectories are initially skewed by gross heading errors. By exchanging their positioning estimates and uncertainties, cooperative navigation helped to refine their trajectories, reduce their position error by at least 55 percent, and provide up to one-meter accuracy [4]. Incorporating this technique provided the desired level of accuracy and maintained it throughout the duration of their walk along the closed-loop path. A PNS that incorporates cooperative navigation, radio-based ranging and foot-mounted IMU's may not be the final solution employed in this research, but these technologies and techniques demonstrate potential for solving the indoor positioning problem with less than one meter of error. A PNS can be developed to track riflemen's positions for this specific application.

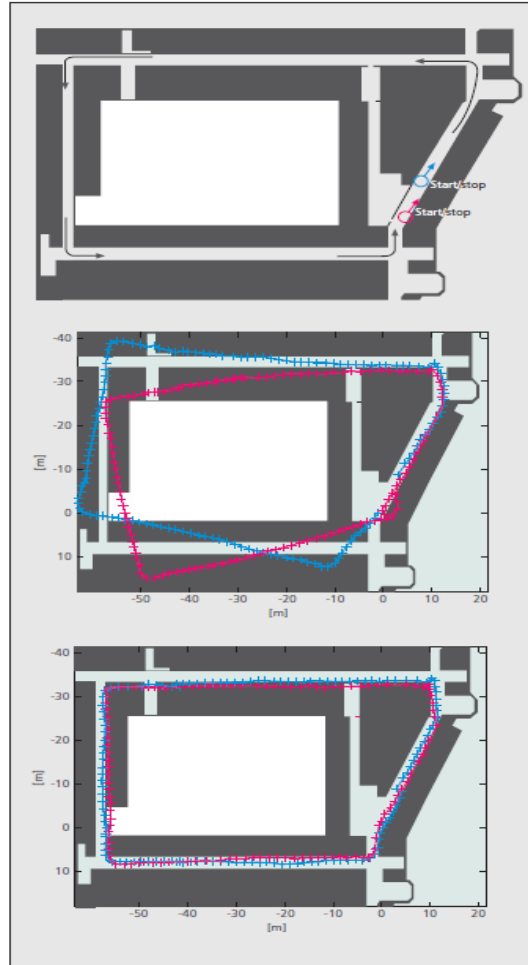


Figure 3. Two units use cooperative navigation to refine their positions as they walk around a closed-loop corridor. From [4].

2. Tactical Network

A tactical network must be configured to track and communicate each rifleman's position in an urban environment and aboard a ship. This network must be able to exchange data between individual riflemen similarly to the way Blue Force Tracker connects mobile platforms [5]. It must be a lightweight and affordable design that can be integrated at the platoon-level. CQC operations require detailed coordination at this unit size and level of command, but these operations are in environments that can restrict a network's reach. Ideally, it is designed as an ad-hoc network that can scale its size to expand and contract as a unit's dispersion changes throughout an operation. In addition to the PNS equipment, this network requires each rifleman to be equipped with a communications device. This device allows the riflemen's positions to be tracked and shared by the network and publishes the positional data to the other network subscribers. A block diagram of this data flow is shown in Figure 4. Current thesis research at NPS demonstrates potential for developing a tactical network for this specific application. A scaled-down version for a fire team was created with XBee Pro radios, Arduino Uno microcontrollers, Raspberry Pi computers, and the Robot Operating System (ROS) [5].

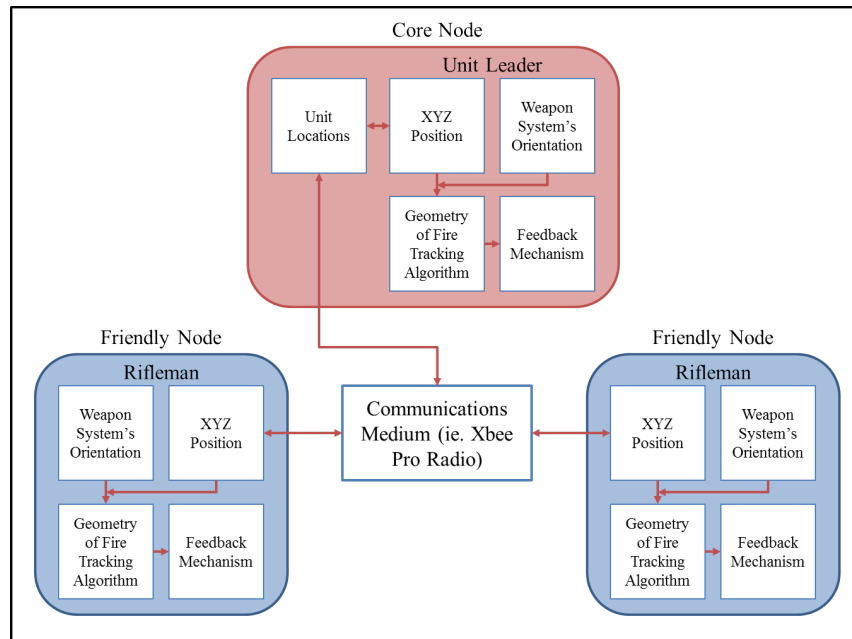


Figure 4. A block diagram that depicts the data flow between the network's core and the riflemen nodes. After [5].

3. CQC Considerations

CQC is an extremely dynamic type of combat that is fought in extremely challenging environments. Friendly units can be isolated and contained in confined spaces on a ship or in an urban development. The close proximities in these environments force friendly units into small-unit engagements with the enemy. As such, senior leadership must relinquish control to their small unit leaders and must rely on the initiative of their subordinates to accomplish the mission. Fire teams serve as the building blocks of an infantry unit. In CQC, these four-man teams often operate independently of their adjacent and higher units. Their speed, aggression, and cunning are required to defeat the enemy in these small-unit engagements [6]. Since CQC demands decentralized control at this level, fire team leaders are responsible for maintaining their unit's geometries. They must maintain proper offsets as their fire teams fire and move. Close proximities and fire teams are the major considerations that impact the system's overall design. If a basic prototype can be validated for a fire team operating in confined spaces, then it provides a viable model that can be expanded for larger units dispersed over larger areas.

The close proximities in CQC require the system to identify geometry issues as accurately and quickly as possible. The sensors that were selected and the algorithms that were designed should be optimized for these two variables. Their level of accuracy and speed should be maintained for dynamic conditions. In these operations, riflemen rarely maintain fixed positions and orientations. They constantly move and reorient with their fire team. The system must be able to track these changes as accurately as possible in real-time. Additionally, the system must inform riflemen when geometries are conflicted without any actions required by the riflemen. This improves their situational awareness and enables them to quickly decide whether to employ their weapon systems. If actions are required by the riflemen, the system does not streamline the geometry-of-fire validation cycle. It only makes it more cumbersome. Lastly, the system must be designed to modularly expand and contract at the fire team level like an infantry unit. As mentioned, fire teams serve as the building blocks of an infantry unit's task organization. A unit can administratively tailor its task organization to a specific mission, and it can

operationally expand and contract its dispersion throughout the mission. As such, the system must be able to follow suit. Its hardware and software design must mirror an infantry unit's administrative task organization and operational dispersion for a CQC operation.

B. CONE OF FIRE

A weapon system's cone of fire determines the safety threshold, or offset, that friendly units must maintain for proper geometries. Cone of fire is a term that describes the trajectory pattern of a round as it is fired. Each round follows a slightly different path due to variations in ammunition, vibrations within the weapon system, and changes in atmospheric conditions [7]. This three-dimensional cone extends from a weapon's muzzle to its beaten zone as shown in Figure 5.

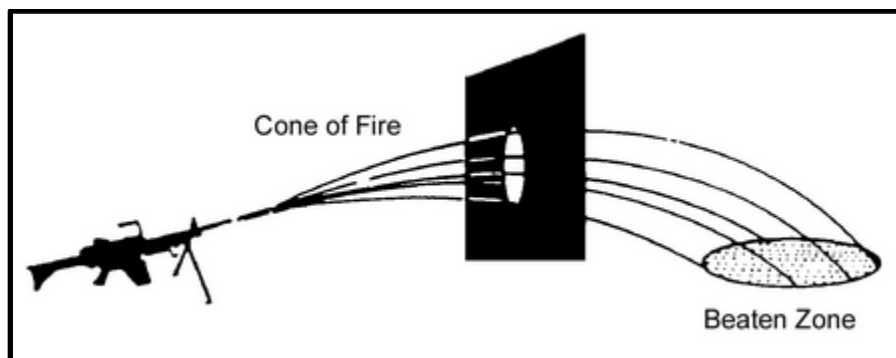


Figure 5. The trajectory pattern of a fired round is described by a cone of fire.
From [7].

Riflemen estimate the size of these cones with angular measurements from their apexes at a weapon's muzzle. These angles are typically measured in mils, where $1 \text{ mil} \approx 0.056 \text{ degree}$. The sizes of these cones and their corresponding safety thresholds depend on the weapon system. As an example, the safety threshold for an M4 carbine's cone of fire is 300 mils, or approximately 16.9 degrees. All friendly positions must maintain an offset of at least 16.9 degrees in any direction for a rifleman to safely engage targets. The remainder of this research focuses on the M4 carbine and its cone of fire. Riflemen are standardly issued an M16 rifle or M4 carbine. The same angular measurement is used for both of these weapon systems' safety thresholds since the M4

carbine is a short-barrel version of the full length M16 rifle. The shorter length of the M4 carbine makes it more desirable for the confined spaces of a CQC operation. A kinematic model of a rifleman and a tracking algorithm for geometry of fires were developed with the M4 carbine and its cone of fire. It is assumed that the model and algorithm can be easily adapted to other direct-fire weapon systems if they are verified for the M4 carbine.

Geometry-of-fire issues occur when a friendly position is located within a rifleman's cone of fire. Currently, riflemen are not provided with any technological advantage that helps them to identify when this occurs. Marine riflemen are trained to rely on the *300 Mil Rule* in the Fleet Marine Force. This antiquated rule is named after the 300-mil cone of fire, and it is a quick reference tool that allows riflemen to visually assess if a friendly position is located within potential trajectories of their rounds. As shown in Figure 6, placing a thumb on the front-sight post of the carbine and extending the pinky finger approximates a 300-mil cone of fire. Any friendly position that can be seen between the front-sight post and extended pinky finger within the M4 carbine's maximum effective range is a geometry-of-fire issue. Riflemen are trained to not engage a target if this occurs. Three major problems are inherent with this rule. First, riflemen's hand sizes are not all the same. This causes inaccurate estimations of these safety thresholds for riflemen whose hands are too big or too small. Second, riflemen must remove a hand from its firing position in order to estimate these angles. This slows riflemen's reaction time for engaging targets in a type of combat that demands minimal hesitation. Third, riflemen cannot always see the locations of other friendly positions in an urban environment or aboard a ship. Walls, bulkheads, and other structures can conceal friendly units from direct line-of-sight even though these structures may not protect them from a rifleman's weapon effects.

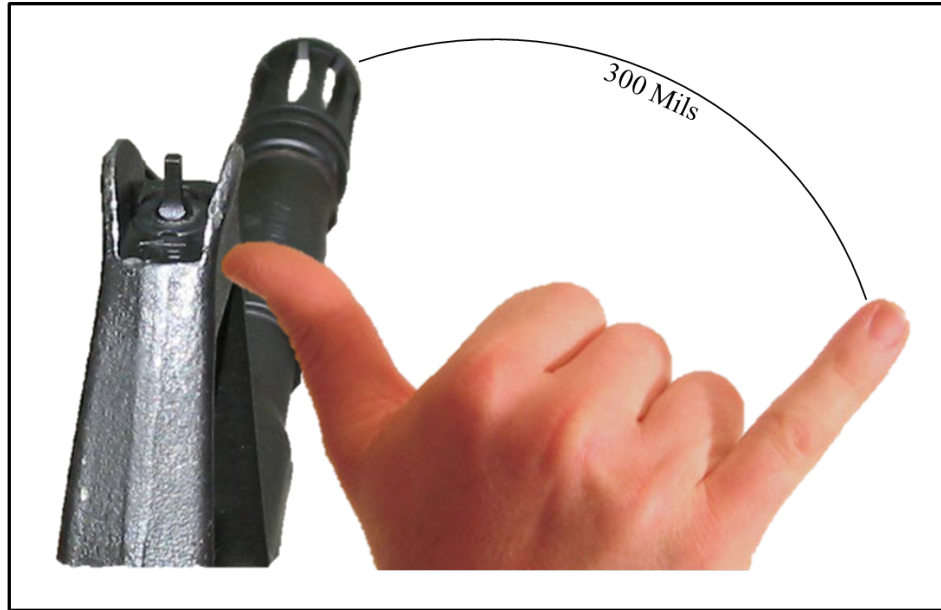


Figure 6. A rifleman implements the *300 Mil Rule* by placing a thumb on the front-sight post and extending the pinky finger. After [8], [9].

This research attempts to remedy the inherent problems of the *300 Mil Rule*. The goal is to provide a technological advantage to riflemen that identifies when geometries are conflicted with more precision, without any actions required by the riflemen, and without the need for direct line-of-sight. An orientation estimation sensor can help to accomplish this by tracking the pose of each rifleman's weapon system. Offsets between friendly nodes in a tactical network can then be computed with the riflemen's positions from the PNS and their weapon systems' poses from these orientation estimation sensors. The precision of this method is limited by the accuracy of the PNS and the orientation estimation sensors. No actions are required by the rifleman, and direct line-of-sight is not needed. A feedback mechanism, like a light in the optic, vibration in the buttstock, or audible tone in a headset, can alert the riflemen of any geometries issues, and these offsets can be computed between any friendly nodes in the network regardless of whether they are directly visible to each other.

The geometries between friendly nodes can be represented by vectors. These vectors are three-dimensional in order to represent the three-dimensions of a battlefield. As shown in Figure 7, a vector $\vec{W}$ is used to describe the orientation of a rifleman's

weapon system, and a vector $\vec{F}$ is used to describe a friendly's position relative to the rifleman. An angle ϕ is derived from the *300 Mil Rule* to represent the safety threshold that surrounds a weapon system's cone of fire, and the angle θ between $\vec{F}$ and $\vec{W}$ is compared to ϕ in order to identify the potential for friendly fire. A geometry-of-fire issue exists if $|\theta| \leq \phi$. Two scenarios are depicted with top-down views in Figure 7. In both scenarios, the riflemen are located at the apex of the cones of fire with their weapon systems oriented straight ahead. The cones of fire are centered along $\vec{W}$ with ϕ on either side. A geometry-of-fire issue is not present in the scenario on the left since $|\theta| > \phi$. The rifleman in this scenario can safely engage targets. In the scenario on the right, a geometry-of-fire issue is present since $|\theta| \leq \phi$, and the rifleman must not engage any targets.

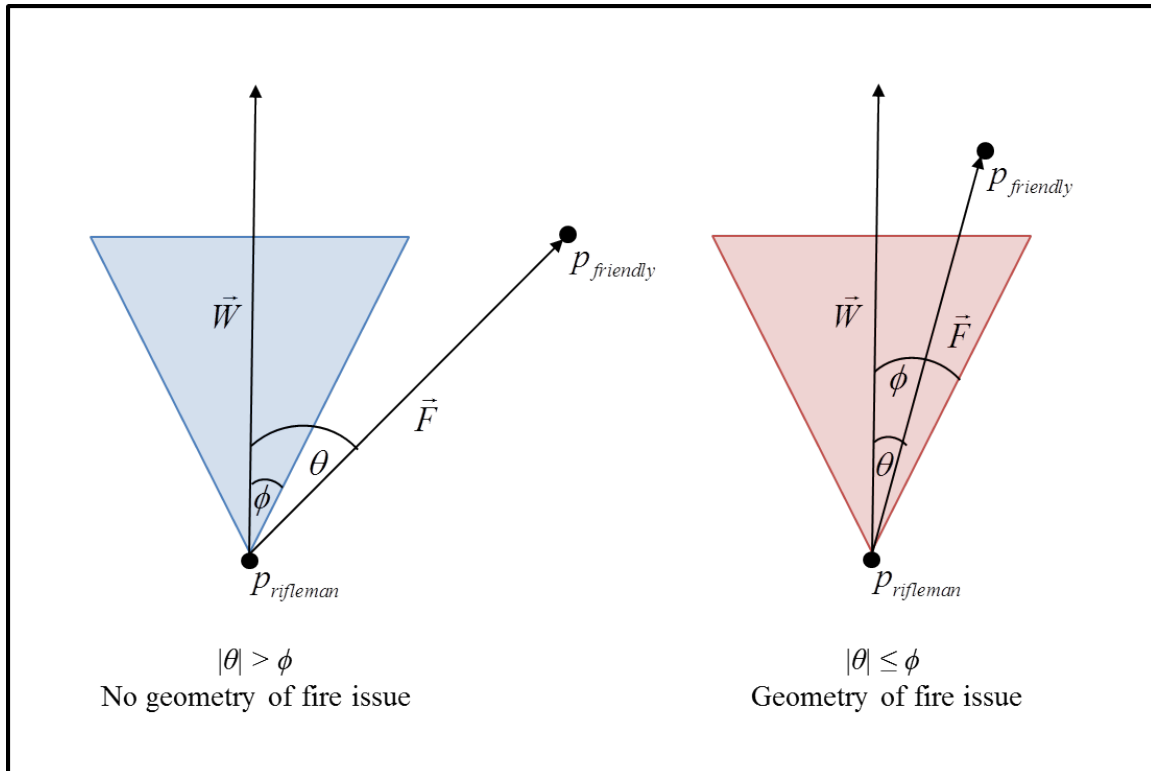


Figure 7. Two-dimensional representation of the vectors and angles that are used to determine if a geometry-of-fire issue exists.

Let the rifleman's positions $p_{rifleman}$ serve as the origin of a three-dimensional Cartesian coordinate system. The vector $\vec{W}$ extends 500 meters from this origin to the M4 carbine's maximum effective range. Its direction is determined by the orientation estimation sensor. The vector $\vec{F}$ is computed in Cartesian coordinates with $\vec{F} = p_{friendly} - p_{rifleman}$. The friendly's position $p_{friendly}$ is determined by the PNS and published by the tactical network to the rifleman's node. The angle ϕ for the cone of fire is a known constant that depends on the rifleman's weapon system. For an M4 carbine, its constant angle $\phi = 16.9$ degrees. The angle θ between vectors $\vec{F}$ and $\vec{W}$ must be computed. After converting $\vec{W}$ into Cartesian coordinates, we compute the angle θ with

$$\theta = \arctan\left(\frac{\|\vec{F} \times \vec{W}\|}{\vec{F} \cdot \vec{W}}\right). \quad (2.1)$$

All of these variables, except the known constant for ϕ , depend on time. These positions, vectors, and angles must be tracked and updated by the tactical network as the riflemen move their positions and orientations on a battlefield. A reference frame is required to efficiently track these changing variables. Each rifleman's position cannot serve as the origin to its own reference frame in the network. Tracking all of these variables in different coordinate systems is too cumbersome, especially for a large network. Instead, these variables can be tracked in a single common frame. Spatial descriptions and transformations were investigated to further develop this geometry-of-fire tracking algorithm. Transformations of these variables were explored, and the kinematics of a rifleman was modeled.

C. ALGORITHMS

1. Reference Coordinate System

Coordinate systems are used to uniquely describe riflemen's positions and the orientations of their weapon systems. Multiple systems are attached to the rigid bodies of a rifleman and weapon system in order to properly model their kinematics. The description of their bodies' positions and orientations can be transformed from one

coordinate system to another. This allows for any of these systems to serve as the reference frame for the bodies' kinematics [10]; however, it is important to select a universal system to serve as a common reference [11]. This makes the transformations from body to body as convenient as possible. Since this research focuses on tactical scenarios in localized areas, the North-East-Down (NED) coordinate system is an appropriate reference frame for this application. It serves as an inertial system with its x -axis aligned with earth's magnetic north, y -axis aligned with east and z -axis pointing straight down. All of the positions, vectors and angles can be described relative to the NED coordinate system, allowing the tactical network to efficiently track changes on a battlefield in a single common reference frame. Transformation operators were defined for a rifleman's kinematic model in order to describe these variables with respect to the NED coordinate system.

2. Model of a Rifleman's Kinematics

Kinematics is a science of motion. It ignores the forces that actually cause motion and focuses on the study of position and its higher-order derivatives with respect to time [10]. It allows the geometrical properties of a rigid body to be described as the body moves [10]. In order to fully develop a tracking algorithm, the kinematics of a rifleman must be modeled. The goal is to describe the position and orientation of a weapon system as it is manipulated by a rifleman. For this model, an M4 carbine mounted to a tripod represents a direct-fire weapon system being carried by a rifleman. The tripod simplifies the kinematics of the model by reducing the number of degrees of freedom and allows the M4 carbine to be held in fixed positions and orientations. A rifleman cannot hold a weapon system perfectly steady. This induces human error into the model of the weapon system's positions and orientations. Removing this human error with the tripod helps with the verification of the kinematic model and the validation of the orientation estimation sensor's accuracy. A depiction of the M4 mounted to a tripod is shown in Figure 8.

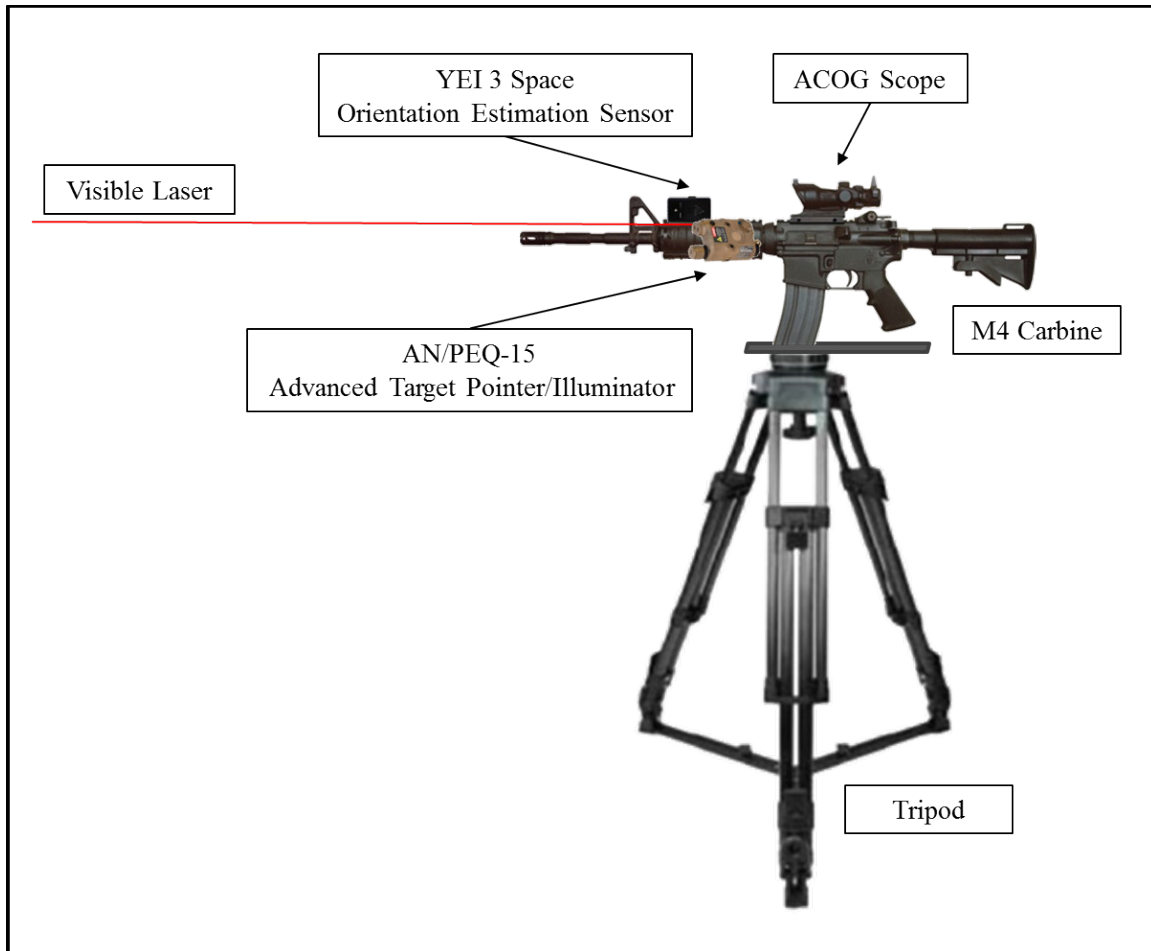


Figure 8. An M4 carbine mounted to a tripod is used to model a rifleman's kinematics. After [12]–[15].

Four coordinate systems are attached to this configuration. The weapon system and tripod each require two systems in order to properly model their translations and rotations. Their corresponding systems are shown in Figure 9. The weapon system consists of the M4 carbine, Advanced Combat Optical Gunsight (ACOG), AN/PEQ-15 Advanced Target Pointer/Illuminator (PEQ-15), and YEI 3 Space orientation estimation sensor. Its $\{L\}$ coordinate system for the PEQ-15 and $\{S\}$ coordinate system for the sensor can completely describe its kinematics. It is standard practice in the military to bore-sight the PEQ-15's visible laser to the carbine's barrel and aim point of the ACOG's reticle. Bore-sighting aligns the laser, barrel, and reticle's aim point and allows the laser to serve as an auxiliary aim point in CQC. In close proximities, riflemen are trained to

keep their weapon system in a ready position and to engage targets while aiming with the PEQ-15's laser. This reduces the amount of time that it takes to engage a target since riflemen can look over their scopes and do not need to sight-in with their ACOG's reticle. The assumption is that their weapon systems are oriented where their lasers point. As such, the weapon system does not require three separate coordinate systems for each of these components. Its orientation can be modeled with $\{L\}$ only. Even though it is also fixed to the carbine's rigid body, the sensor requires its own coordinate system in order to track the weapon system's orientation. Its axes may not perfectly align with $\{L\}$ as it is mounted; therefore, its coordinate system $\{S\}$ is needed to negate this initial rotational offset.

The tripod has two joints that are physically offset and allow for its yaw and pitch rotations. These joints require separate coordinate systems in order to properly model the tripod's rotations. The bottom joint allows for its yaw rotations, or rotations from left to right, and the top joint allows for its pitch rotations, or rotations up and down. The $\{Y\}$ and $\{P\}$ coordinate systems are attached, respectively, to these joints. The tripod does not allow for roll rotations, or rotations about the axis of the weapon system's barrel. This limitation is appropriate since it is not practical to change a weapon system's aim point with a roll. Doctrinal firing positions ideally have no rotation about the x -axes, or axes along the weapon system's barrel. The z -axes should remain perpendicular to the ground, and the y -axes should remain parallel to the ground. Riflemen are trained to implement yaws and pitches to change their weapon system's aim point. Since rolling is not practically desirable, ignoring these rotations is justifiable for this model. As a result, the kinematics of this configuration can be completely modeled with the $\{L\}$, $\{S\}$, $\{Y\}$, and $\{P\}$ coordinate systems.

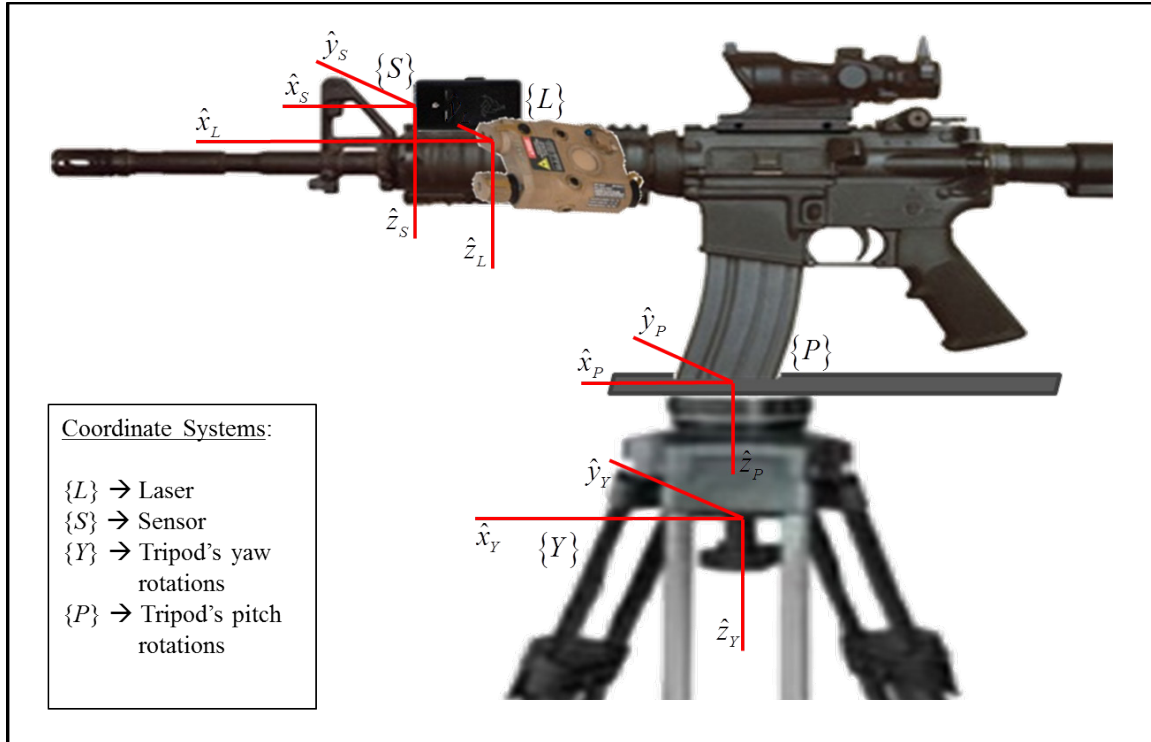


Figure 9. Four coordinate systems are attached to the weapon system and tripod for the kinematic model. After [12]–[15].

The translations and rotations from each coordinate system can be described with transformation operators. These operators are 4×4 matrices that represent homogeneous transforms and represent functions that map a description from one coordinate system to another [11]. This links the systems together by mapping the rotations and translations between their coordinate descriptions. These operators incorporate an orthonormal 3×3 matrix that maps the rotations and a 3×1 vector that maps the translations. The translation vectors describe the position of a coordinate system's origin with respect to another. This links the origins of each system together and ultimately allows the position of each coordinate system to be described relative to the NED reference frame through compound transformations. These translation vectors are shown in Figure 10.

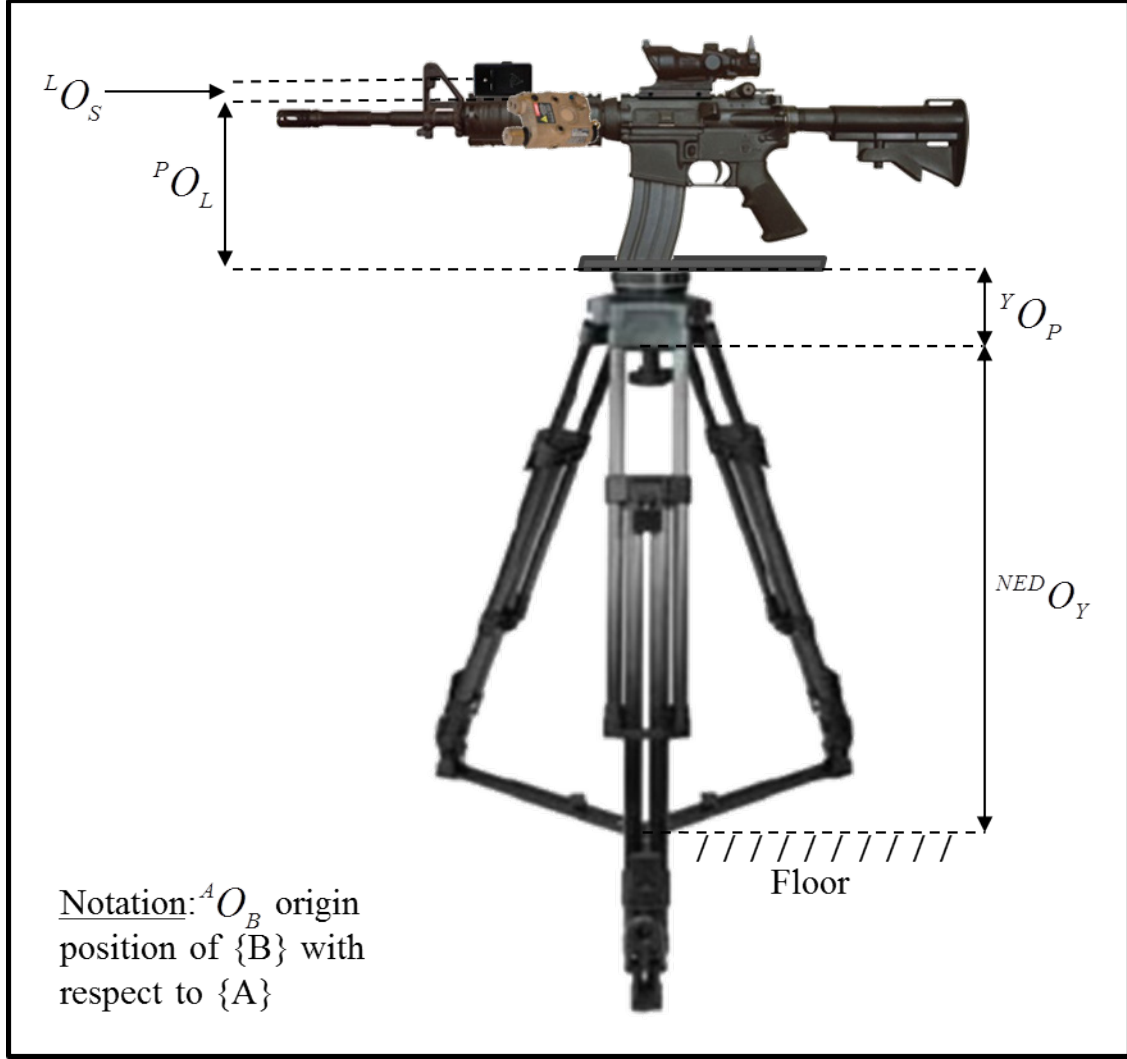


Figure 10. Translation vectors between the four coordinate systems that are attached to the weapon system and tripod. After [12]–[15].

The general form of a transformation operator for two arbitrary coordinate systems $\{A\}$ and $\{B\}$ is

$${}^A T_B = \begin{bmatrix} {}^A R_B & {}^A O_B \\ 0 & 1 \end{bmatrix}. \quad (2.2)$$

The rotation matrix ${}^A R_B$ describes the rotation of $\{B\}$ relative to $\{A\}$ with Euler angles, and the translation vector ${}^A O_B$ describes the position of $\{B\}$'s origin relative to $\{A\}$ with three-dimensional Cartesian coordinates [11]. The yaw, pitch, and roll angles from the

rotation matrix are measured in degrees and denoted with α , β , and γ , respectively. The Cartesian coordinates from the translation vector are measured in meters and denoted with ${}^A x_B$, ${}^A y_B$, and ${}^A z_B$. These operators were defined for each pair of coordinate systems that are physically linked in the rifleman's kinematic model.

This configuration requires four transformation operators ${}^{NED}_y T$, ${}_p^y T$, ${}_L^p T$, and ${}_s^L T$. The operator ${}^{NED}_y T$ describes the tripod's yaw rotations with respect to the NED reference frame and the height of the tripod's bottom joint from the floor. The operator ${}_p^y T$ describes the tripod's pitch rotations with respect to the NED reference frame and the offset between the tripod's top and bottom joints. The operator ${}_L^p T$ does not account for any rotations since the weapon system is fixed to the mounting plate on the tripod's top joint. Its rotation matrix is an identity matrix. This operator only describes the offset between the PEQ-15 and the tripod's top joint. Similarly, the operator ${}_s^L T$ only describes the offset between the sensor and the PEQ-15. These operators are defined as

$${}^{NED}_y T = \begin{bmatrix} -\cos(\alpha) & \sin(\alpha) & 0 & {}^{NED}x_y \\ -\sin(\alpha) & -\cos(\alpha) & 0 & {}^{NED}y_y \\ 0 & 0 & 1 & {}^{NED}z_y \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.3)$$

$${}_p^y T = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & {}^y x_p \\ 0 & 1 & 0 & {}^y y_p \\ -\sin(\beta) & 0 & \cos(\beta) & {}^y z_p \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.4)$$

$${}_L^p T = \begin{bmatrix} 1 & 0 & 0 & {}^p x_L \\ 0 & 1 & 0 & {}^p y_L \\ 0 & 0 & 1 & {}^p z_L \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.5)$$

and

$${}^L_sT = \begin{bmatrix} 1 & 0 & 0 & {}^Lx_s \\ 0 & 1 & 0 & {}^Ly_s \\ 0 & 0 & 1 & {}^Lz_s \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.6)$$

Since the weapon system is assumed to be oriented where the PEQ-15's laser is aiming, these transformation operators are able to describe the PEQ-15's position and orientation relative to the NED reference frame. Even though the PEQ-15's coordinate system $\{L\}$ is not directly linked to $\{NED\}$, it can still be described relative to the reference frame through compound transformations. Compound transformations chain these operators together through matrix multiplication [11]. The operator ${}^{NED}_LT$ describes the weapon system's position and orientation relative to the NED reference frame and is computed with

$${}^{NED}_LT = {}^{NED}_YT {}^Y_P T {}^P_L T. \quad (2.7)$$

These five transformation operators complete the model of a rifleman's kinematics. Four coordinate systems were attached to the M4 carbine mounted to a tripod. The five operators link these systems together and describe the weapon system's position and orientation with respect to the NED reference frame. This kinematic model needs to be verified before it is implemented in the geometry-of-fire tracking algorithm. An experiment was designed to compare the model's description of the weapon system with ground-truth data, and this design is addressed in the next chapter. Refinement of this model is also recommended for future work. The tripod from this configuration must eventually be replaced by an actual human body. Modeling a human body's degrees of freedom is challenging but is critical for describing a weapon system's kinematics as a rifleman manipulates it during an actual CQC operation.

3. Geometry-of-Fire Tracking Algorithm

The algorithm for tracking geometry of fires must be able to identify when a friendly position is located within a rifleman's cone of fire. This algorithm is separated

into three main steps. First, a reference orientation for the weapon system is determined from the sensor's initial readings. Second, the weapon system's orientation is tracked as it is manipulated throughout a CQC operation. Third, the geometries between friendly nodes are computed as the weapon system's orientation and friendly positions change. As discussed, a geometry-of-fire issue exists if $|\theta| \leq \phi$. A flow chart of this algorithm is shown in Figure 11.

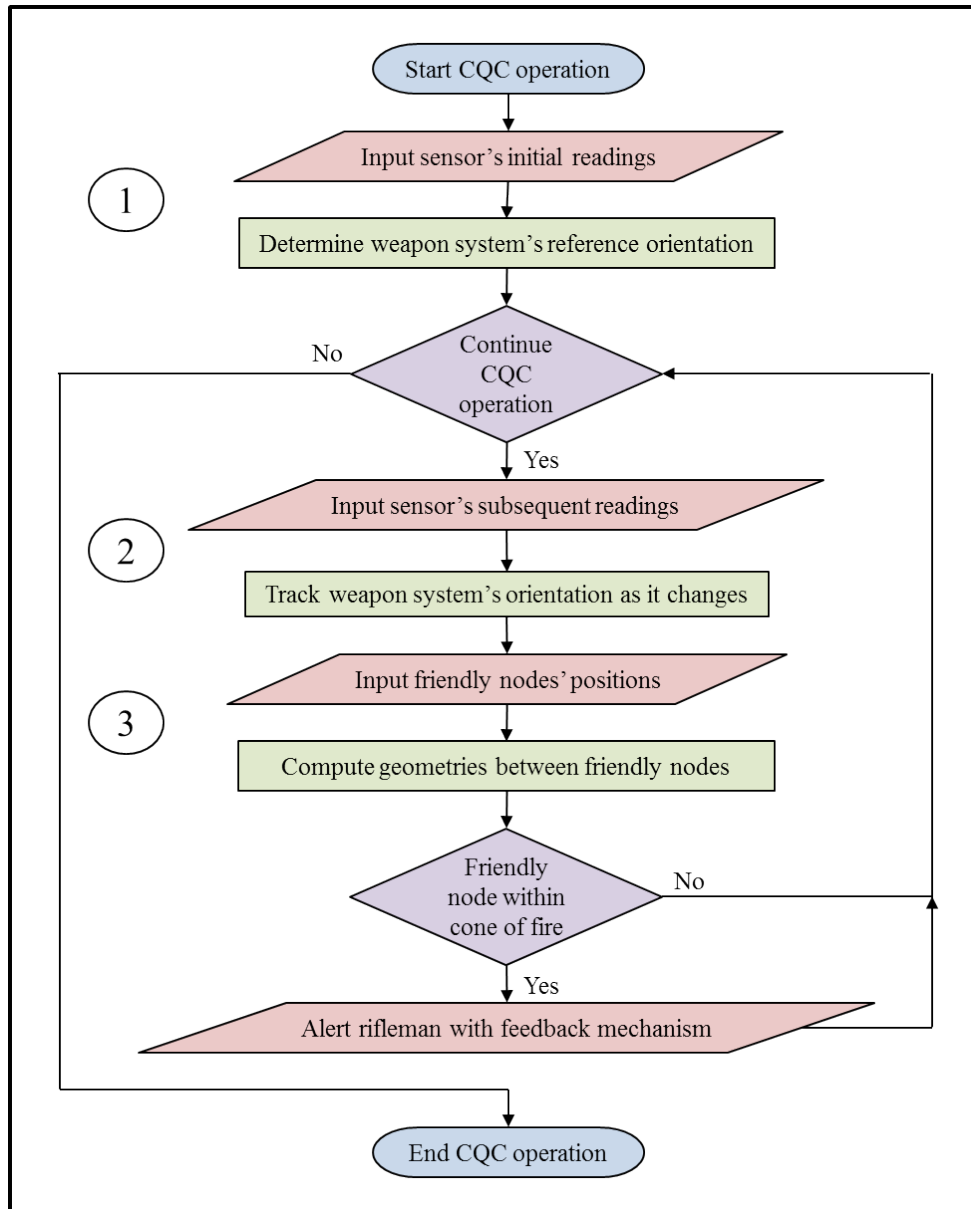


Figure 11. The main steps of the geometry-of-fire tracking algorithm.

The sensor provides estimated orientations of the weapon system with respect to $\{S\}$. Since the sensor's axes rarely align with the NED reference frame at the start of a CQC operation, a reference orientation is determined to describe their initial rotational offset. This reference orientation allows the misalignment between $\{S\}$ and $\{NED\}$ to be negated and a rifleman to carry their weapon system in any pose at the start of an operation. All subsequent orientations are described relative to this reference. In order to determine this reference orientation, the weapon system is held in a fixed pose. An initial set of quaternions is collected from the sensor to estimate the weapon system's starting orientation with respect to the NED reference frame. These quaternions are converted into a rotation matrix. This matrix describes the weapon system's reference orientation and its initial offset to the NED reference frame. This rotation matrix ${}^{NED}_{S_o}R$ is computed from the mean quaternion of the set with

$${}^{NED}_{S_o}R = \begin{bmatrix} 2q_0^2 - 1 + 2q_1^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 2q_0^2 - 1 + 2q_2^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 2q_0^2 - 1 + 2q_3^2 \end{bmatrix}, \quad (2.8)$$

where the quaternion q is defined as the sum $q = q_0 + iq_1 + jq_2 + kq_3$ and i , j , and k represent the standard orthonormal basis of a three-dimensional space [16].

The weapon system's pose can be tracked throughout an operation once this reference orientation is determined. As the weapon system is manipulated, the sensor continues to collect quaternions. These quaternions are converted into rotations matrices with the same computations that are used for the reference orientation. The rotation matrix ${}^{NED}_{S_i}R$ describes these subsequent orientations of the weapon system. This matrix is updated for each quaternion, and compound transformations can describe these subsequent orientations with respect to the reference orientation. It is important to note that the orthonormal properties of these rotation matrices allow for the descriptions of these matrices to be easily inverted. Transposing ${}^{NED}_{S_o}R$ computes the rotation matrix ${}^{S_o}_{NED}R$ that describes the NED reference frame relative to the weapon system's reference orientation [11]. This inverted description is needed to determine the rotation matrix ${}^{S_o}_{S_i}R$

that describes the subsequent orientations of the weapon system with respect to its reference orientation. This rotation matrix ${}^{S_o}_{S_i}R$ is computed with

$${}^{S_o}_{S_i}R = \left({}^{NED}_{S_o}R \right)^T {}^{NED}_{S_i}R. \quad (2.9)$$

The Euler angles that describe these subsequent rotations are determined from this matrix.

These Euler angles are needed to compute the special homogeneous transformation operators that track the weapon system's changing orientation. The standard Z-Y-X angle convention is used for this algorithm [11]. The special operators $T_Z(\alpha)$ and $T_Y(\beta)$ are computed from these angles. These operators describe the weapon system's yaw and pitch rotations, respectively, as a rifleman manipulates the weapon system. They are incorporated into (2.7) in order to track these rotations about $\{L\}$'s fixed axes. The compound transformation equation for ${}^{NED}_LT$ is modified to

$${}^{NED}_LT = {}^{NED}_YT \left(T_Z(\alpha) {}^Y_P T \right) \left(T_Y(\beta) {}^P_L T \right). \quad (2.10)$$

The Euler angles that are required for these two special operators are determined from ${}^{S_o}_{S_i}R$. Given

$${}^{S_o}_{S_i}R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad (2.11)$$

the angles α and β can be computed with

$$\begin{aligned} \alpha &= \arctan 2(r_{23} / \sin \beta, r_{13} / \sin \beta) \\ \beta &= \arctan 2\left(\sqrt{r_{31}^2 + r_{32}^2}, r_{33}\right). \end{aligned} \quad (2.12)$$

These equations bound the pitch angle to $0^\circ \leq \beta \leq 180^\circ$. It is important to note that the solution for α does not exist when $\beta = 0^\circ$ or $\beta = 180^\circ$. When this is the case, the angle α is set to zero [11].

The geometries between friendly nodes can now be computed from the modified ${}^{NED}_L T$. As detailed in the cone of fire discussion, vectors represent these geometries, and the angle θ between two friendly nodes is computed with (2.1). If $|\theta| \leq \phi$, then a friendly node is located within the weapon system's cone of fire, and the rifleman must be alerted by a feedback mechanism that a geometry-of-fire issue exists. Again, the vector $\vec{W}$ extends from the weapon system's muzzle to its maximum effective range along the x -axis of $\{L\}$. This vector is computed for each modification to ${}^{NED}_L T$. If $\vec{W}_o$ represents this vector at the weapon system's reference orientation, then $\vec{W}$ can be computed with

$$\vec{W} = {}^{NED}_L T \vec{W}_o. \quad (2.13)$$

This vector $\vec{W}$ is compared to $\vec{F}$ with (2.1) for every friendly node in the network. This comparison is conducted for the duration of the operation as the weapon system's orientation and friendly positions change. Once the operation is completed, this algorithm terminates. An experiment was designed to test that this algorithm properly detects geometry-of-fires issues. A major consideration that needs to be addressed is when this comparison should be conducted. Should this loop be time or event-based? If it should be event-based, should it be executed when the weapon system's orientation changes, when the friendly positions change or when either of these events change? This experiment's design and consideration is addressed in the next chapter.

D. SUMMARY

A common understanding of the background knowledge required for this research's experimental design was provided in this chapter. Essential assumptions were examined and the problem for tracking a weapon system's orientations and computing geometries between friendly nodes was framed. Even though the technology for an inertial PNS does not currently exist, the potential for it to be developed was shown with foot-mounted IMU's, radio-based ranging, and cooperative navigation. Current thesis research at NPS also demonstrates the potential for developing a tactical network that tracks and communicates this positional data in an urban environment and aboard a ship.

This system's design is impacted by CQC considerations that narrow its scope to fire teams operating at the tactical level and in localized areas. This allows the NED coordinate system to serve as an inertial reference frame for the kinematic model and tracking algorithm that were introduced. This model and algorithm were verified by experiments that are discussed in the next chapter. Once they are integrated into the overall system's design, they will help to replace the antiquated *300 Mil Rule* with a technology that streamlines the continuous validation cycle for geometry of fires.

THIS PAGE INTENTIONALLY LEFT BLANK

III. EXPERIMENTAL DESIGN

The purpose of this chapter is to detail the experiments that were conducted for this research. As discussed in Chapter I, two elements of the tactical network are focused upon in this thesis: tracking the weapon system's orientation and computing the geometries between friendly nodes. An experiment was conducted for each element in order to verify its design for the system's prototype. The rifleman's kinematic model was verified for tracking the weapon system's orientation, and the geometry-of-fire tracking algorithm was tested for computing the geometries between friendly nodes. Before conducting these experiments, the sensor's accuracy for estimating orientations needed to be validated. The hardware components that were selected for these experiments are discussed and the test plans that validated the sensor's accuracy, verified the kinematic model, and tested the tracking algorithm are detailed in this chapter.

A. HARDWARE COMPONENTS

Two hardware components are discussed before detailing the test plans: the YEI 3-Space Data-Logging sensor that was selected to track the weapon system's orientations and the VICON motion system that was used to provide ground-truth data. The product overviews of these components and why they were chosen for these experiments are summarized in this section.

1. YEI 3-Space Data-Logging Sensor

The 3-Space Data-Logging sensor from YEI Technology was selected to track the weapon systems' orientations for the network. It was mounted to the weapon system as shown in Figure 8 and collected the quaternions that were discussed in the geometry-of-fire tracking algorithm. A technical brief published by YEI Technology states that their miniature sensor is a highly precise and reliable IMU [17]. It uses proprietary quaternion-based Kalman filtering algorithms to estimate orientation from its gyroscope, accelerometer, and magnetic measurements [17]. This estimation can be determined relative to a desired coordinate system, and the sensor ensures optimal precision and accuracy by incorporating confidence intervals into its algorithms [17]. Based on its

product specifications, the IMU's orientation estimations are accurate within $\pm 1^\circ$ for dynamic conditions and all orientations [17]. The 3-Space Data-Logging sensor and its dimensions are shown in Figure 12. With the rapid prototyping mindset, this IMU was selected without further analysis or comparison to other like devices. It claimed enough accuracy for this geometry-of-fires problem, and it was designed to return quaternions that have been filtered and are relative to a specifiable coordinate system. Its level of accuracy needed to be verified, but, otherwise, it met all of the initial requirements off-the-shelf and was readily available in the Department of Electrical and Computer Engineering's Control Systems Laboratory at NPS.

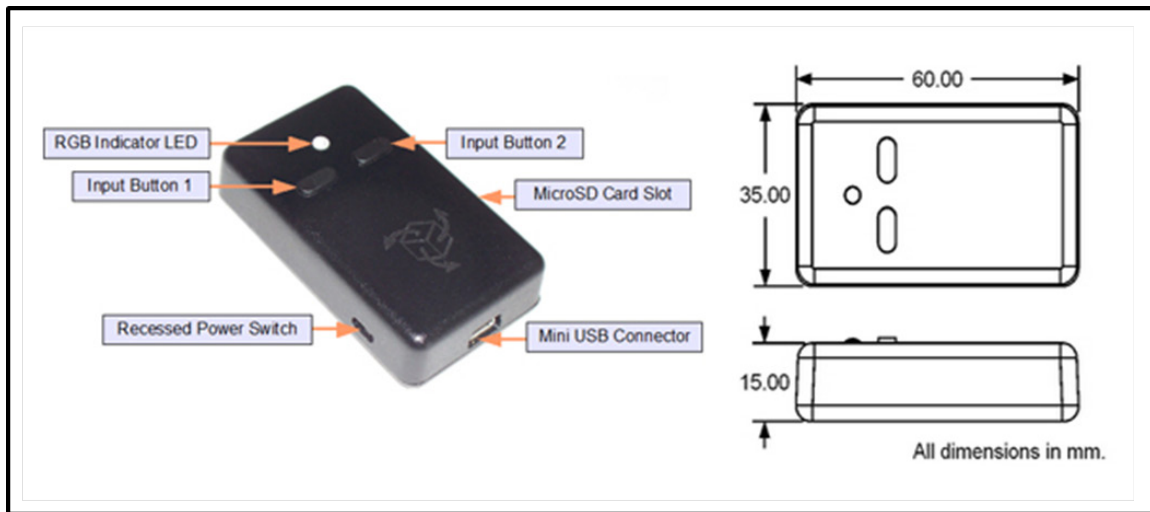


Figure 12. The 3-Space Data-Logging sensor from YEI Technology tracks the weapon systems' orientations for the network. From [15].

2. VICON Motion System

The VICON motion system was selected to provide ground-truth data for the experiments discussed in this thesis. As shown in Figure 13, it is a motion capture system consisting of a camera suite and software interface. This system conducts real-time precision tracking of an object's movements with a technique known as optical-passive motion capturing [18]. This technique uses infrared cameras to track reflective markers that were fixed to the weapon system. The cameras are high-speed and optimized for high-resolution motion capture [18]. This allows the system to track subtle and quick

movements and to describe the position of the reflective markers with millimeter accuracy [19]. It reconstructed the cameras' two-dimensional observations to accurately describe the weapon system's three-dimensional motion. The VICON motion system is marketed for a wide variety of applications from entertainment to engineering. For this specific application, it served the same purpose as the YEI 3-Space Data-Logging sensor. It tracked the weapon system's orientation for the network and provided the geometry-of-fire tracking algorithm with Euler angles to compute its special homogeneous transformation operators. The weapon system's orientations that were described by the sensor's quaternions were compared to these ground truth orientations in order to validate the sensor's accuracy and verify the rifleman's kinematic model.

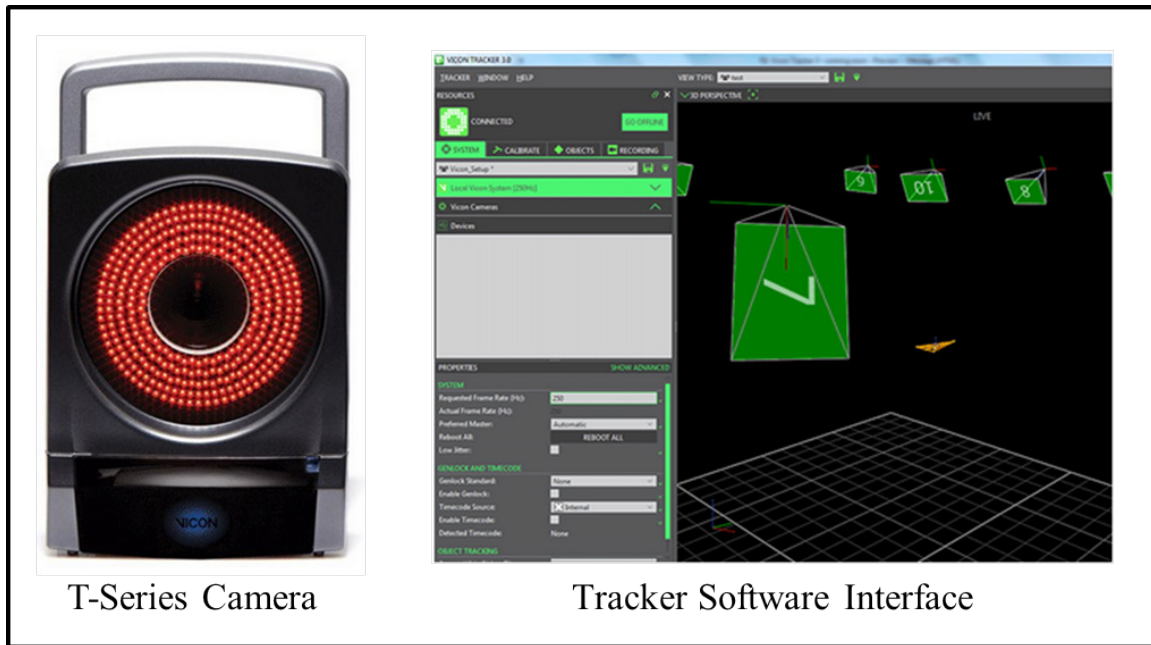


Figure 13. Ground-truth data for the experiments is provided by the VICON motion system. From [18].

B. TEST PLAN

Prototyping the system's design was one of the main objectives of this research. The two elements that serve as this thesis' focus must be validated before they can be integrated into this prototype. As mentioned, three experiments were conducted to evaluate these elements. First, the accuracy of the sensor's orientation estimations was

determined. Second, the model of the rifleman's kinematics was verified for tracking the weapon system's pose. Third, the geometry-of-fire tracking algorithm was tested for computing the geometries between friendly nodes. The test plans that were developed to complete these three experiments are summarized in this section.

1. Validation of the Sensor's Accuracy

The accuracy of the YEI 3-Space Data-Logging sensor needed to be validated before verifying the kinematic model and tracking algorithm. YEI Technology boasts that its IMU's orientation estimations are accurate within $\pm 1^\circ$ [17]. If this level of accuracy is maintained for dynamic conditions and all orientations, then the maximum error for a weapon system's orientation in any direction is 0.18 meters at a ten-meter range and 8.73 meters at a 500-meter range. As discussed with the CQC considerations, a range of ten meters or less are used for this design's proof-of-concept, and the 500-meter range corresponds to the maximum effective range for an M4 carbine. The sensor's maximum errors are acceptable based on the average width of a man's shoulders and the width of an M4 carbine's safety threshold at these ranges. When riflemen conduct their marksmanship qualification ranges, they are taught that the average width of a man's shoulders is 19 inches. Since 0.18 meters is equal to seven inches, only one-third of a man's torso can be errantly within a weapon system's safety threshold at a ten-meter range. At the 500-meter range, an M4 carbine's safety threshold is 152 meters wide, which is much larger than the potential 8.73-meter error. Both of these widths provide sufficient offset from the weapon system's centerline aim point to keep the operational risk at an acceptable level. If geometry-of-fire issues are not detected due to the sensor's maximum errors, friendly nodes still have enough standoff to be safe from a weapon system's munitions effects. This standoff is depicted at both ranges in Figure 14. The desired safety threshold for an M4 carbine's cone of fire is depicted with the red circle, and the standoff that remains with the sensor's maximum error is depicted with the blue circle.

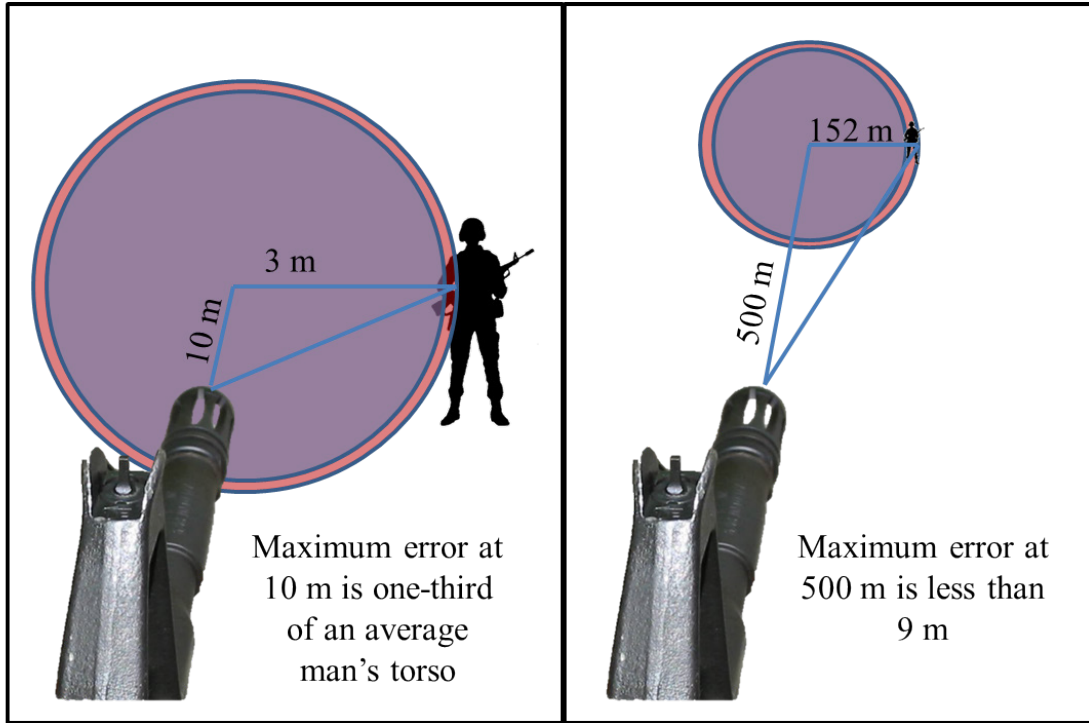


Figure 14. Sufficient standoff is maintained at both ranges despite the sensor's maximum error. After [8].

This level of accuracy needed to be validated for static and dynamic conditions. As such, this experiment was separated into two parts. The sensor's maximum error was determined within the ten-meter range under both of these orientation conditions. It was computed by comparing the sensor's estimations to the VICON system's ground-truth data with (2.1). Vector $\vec{W}$ represents the sensor's estimations, and vector $\vec{F}$ represents the VICON system's ground-truth data. The sensor's angular error was evaluated at various orientations. A two-dimensional Cartesian plane was used to compute this error at precise and repeatable orientations. The weapon system was mounted to the tripod at a fixed position, and this plane was set up orthogonal to $\{L\}$'s x -axis. The weapon system's visible laser pointed at the plane's origin when the weapon system was in its reference orientation. The sensor's error was evaluated for static orientations by collecting data, while the weapon system's visible laser was fixed on grid coordinates in each quadrant of the plane. For dynamic orientations, data was collected while the visible laser was moved along the perimeter of a non-symmetrical polygon that spanned multiple quadrants of the

plane. Given that the weapon system was properly bore sighted, the sensor's accuracy was confirmed if $|\theta| \leq 1^\circ$ for both orientation conditions.

The setup to evaluate the sensor's accuracy for estimating static orientations is depicted in Figure 15.

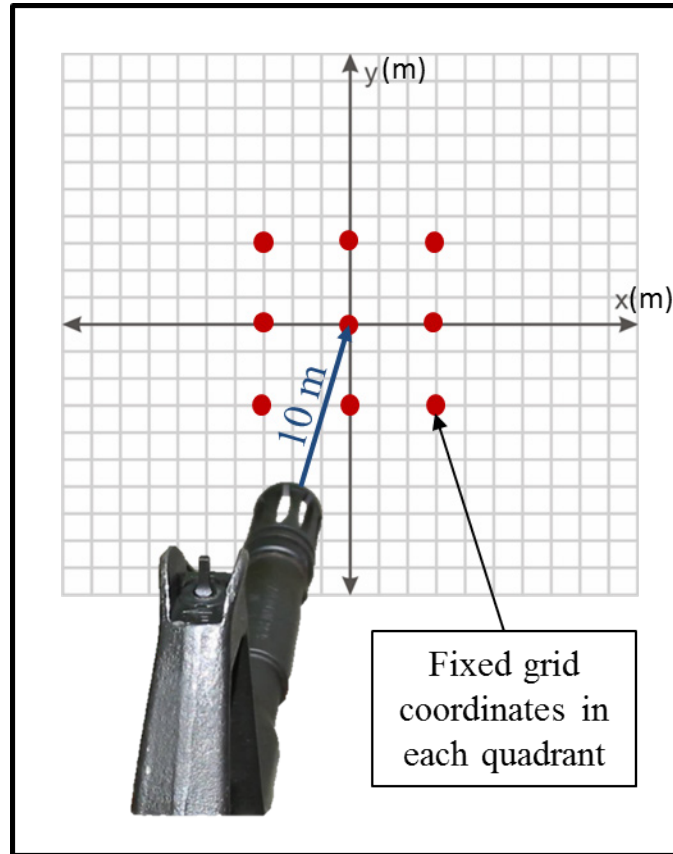


Figure 15. The weapon system's visible laser was held at these various grid coordinates in order to validate the sensor's accuracy for estimating static orientations. After [8].

The setup to evaluate the sensor's accuracy for estimating dynamic orientations is depicted in Figure 16.

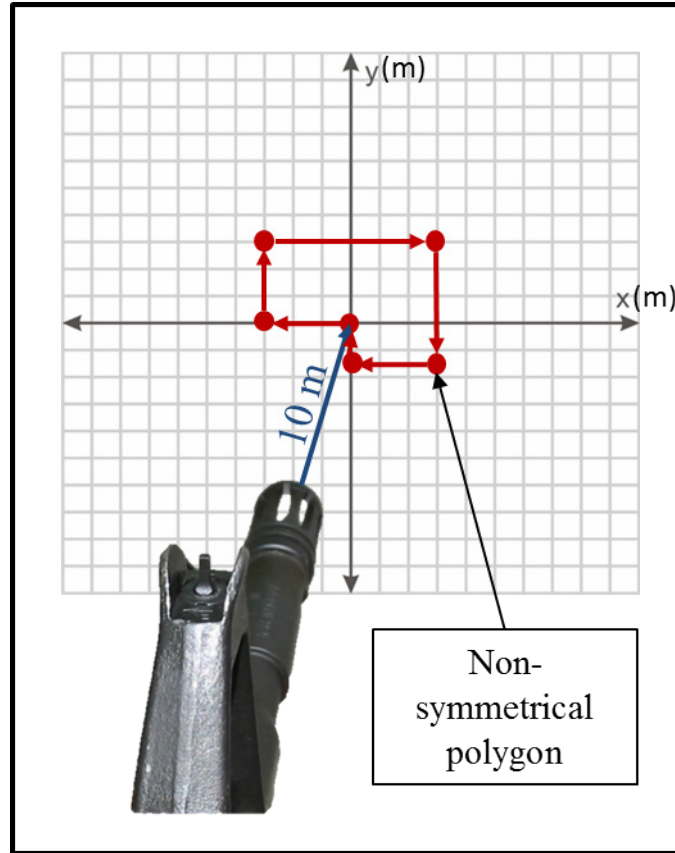


Figure 16. The weapon system's visible laser was moved along the perimeter of this non-symmetrical polygon in order to validate the sensor's accuracy for estimating dynamic orientations. After [8].

2. Verification of the Rifleman's Kinematic Model

The rifleman's kinematic model needed to be verified before it was implemented in the geometry-of-fire tracking algorithm. This experiment compared the model's description of the weapon system to ground-truth data as the weapon system was manipulated. This comparison was conducted with the same setup that is shown in Figure 16. The experiment that was designed for validating the sensor's accuracy under dynamic conditions was repeated, but this repetition evaluated the model's trueness instead of the sensor's accuracy. When the experiment was conducted to evaluate the sensor's accuracy, the objective was to compute the angular difference between $\vec{W}$ and $\vec{F}$. When it was conducted to evaluate the model's trueness, the objective was to compute the angular difference between these vectors and the perimeter of the polygon that was

traced. Given that $\vec{F}$ followed the polygon's perimeter, the kinematic model was confirmed if the sensor's angular difference was less than or equal to its error for dynamic conditions.

Three considerations were taken into account to prevent additional causes of error. Again, the weapon system was properly bore sighted in order to assume that it was oriented where the visible laser was aiming. Additionally, the polygon's perimeter was traced as accurately as possible, and the two-dimensional Cartesian plane was constructed and set up as precisely as possible. Human error was inherent in this experiment since the weapon system was manipulated to trace the polygon's perimeter. Using the tripod mitigated this as much as possible. If the plane was canted or not orthogonal to $\{L\}$'s x -axis, then the orientations of $\vec{W}$ and $\vec{F}$ would have been skewed. The vector $\vec{F}$ was used to see if these considerations caused additional error. If this vector failed to follow the polygon's perimeter, then these considerations were checked before concluding that the kinematic model was not describing the weapon system's motion as desired.

3. Testing of the Geometry-of-Fire Tracking Algorithm

The geometry-of-fire tracking algorithm had to be tested before the elements of this thesis could be integrated into the overall system's prototype. Once the sensor's accuracy was validated and the kinematic model was verified, this experiment ensured that the geometries between friendly nodes were properly computed. The objective for this experiment was to verify that the tracking algorithm properly identifies when geometry-of-fire issues are present. The VICON motion system was no longer required. The kinematic model and tracking algorithm were integrated with the tactical network that was mentioned in Chapter II. Similar to the scaled-down version of this network, a fire team was represented by four nodes constructed with XBee Pro radios, Arduino Uno microcontrollers, Raspberry Pi computers and ROS [5]. The XBee Pro radios and Arduino Uno microcontrollers communicated the positional data of each node as shown in Figure 4, and the Raspberry Pi computers and ROS executed the tracking algorithm and allowed the geometries between the nodes to be computed. One of the nodes was designated as the rifleman, and the remaining nodes were considered three friendly

soldiers. This experiment was conducted in a 10×10 meter area and tested the tracking algorithm for these four situations:

- All nodes at fixed positions
- The rifleman at a fixed position with moving friendly nodes
- The friendly nodes at fixed positions with a moving rifleman
- All nodes moving.

Without an inertial PNS, the positional data of each node was manually fed into the algorithm. Fixed positions were entered manually, and moving positions were simulated. These positions were three-dimensional Cartesian coordinates with respect to the selected NED reference frame. Enclosed in the appendix is the *friendly detection* script file that incorporated this positional data and implemented the tracking algorithm. For the moving positions, the path of each node was predetermined and programmed into this script file. The four situations are depicted in Figure 17.

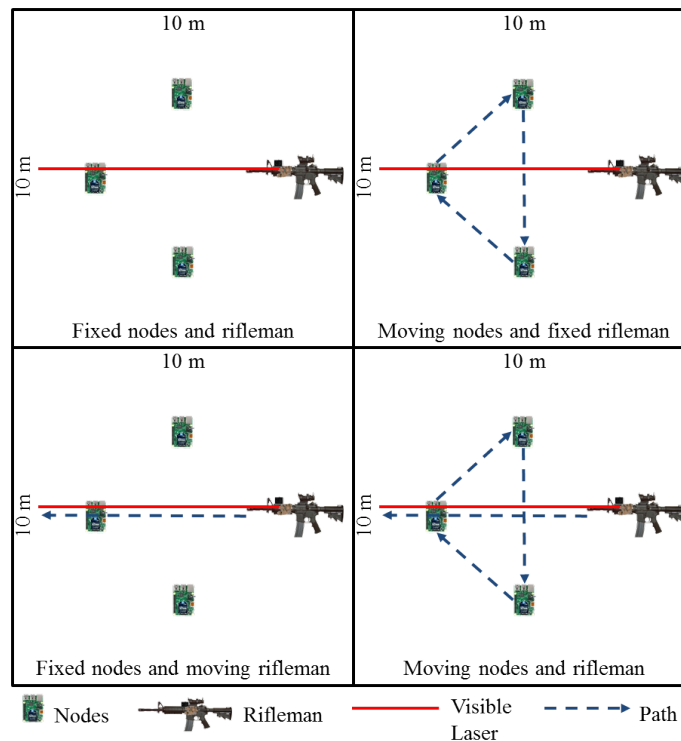


Figure 17. The geometry-of-fire tracking algorithm was tested for these four situations. After [12]–[15].

For each of these situations, the weapon system's orientation was moved from node to node verifying that geometry issues were identified when the laser moved within 16.9 degrees of a node from any direction.

C. SUMMARY

The experiments that were designed for this thesis were detailed in this chapter. Three experiments needed to be conducted before integrating the elements of this thesis into the system's prototype. These experiments validated the accuracy of the YEI 3-Space Data-Logging sensor, verified the rifleman's kinematic model, and tested the geometry-of-fire tracking algorithm. The desired end state was that the tracking algorithm properly identified geometry issues for the four situations that were discussed. The VICON motion system was used to provide ground-truth data in order to evaluate the sensor's accuracy and the model's trueness. The results and analysis from these experiments are discussed in the next chapter. Ideally, the sensor's accuracy remained within $\pm 1^\circ$ for dynamic conditions and all orientations. The kinematic model properly described the weapon system's motion, and the tracking algorithm detected geometry issues when the weapon system's laser moved within 16.9 degrees of a friendly node.

IV. RESULTS AND ANALYSIS

The three experiments that were designed for this thesis build on each other and were conducted sequentially. The sensor needed to accurately estimate the weapon system's orientations in order to verify the rifleman's kinematic model, and the kinematic model needed to properly describe the weapon system's motion in order to test the geometry-of-fire tracking algorithm. As these experiments were conducted, each experiment presented a major challenge. These challenges required adjustments to the original test plans in order to progress from one experiment to the next. The major challenges that were encountered and the results from each experiment are detailed in this chapter.

A. VALIDATION OF THE SENSOR'S ACCURACY

1. Synchronizing the Collected Data Sets

The major challenge for validating the sensor's accuracy under dynamic conditions was synchronizing the collected data sets from the sensor and the VICON motion system. The VICON motion system collected measurements at a different sampling rate and collected measurements for a longer period of time due to the procedural constraints of the experiment. The VICON motion system started collecting measurements before the sensor and stopped collecting measurements after the sensor. These considerations caused the data sets to contain different numbers of samples and prevented a one-to-one comparison of their corresponding orientations. The different number of samples and lack of one-to-one relationship is shown with plots of the Euler angles from the sensor and VICON motion system in Figure 18.

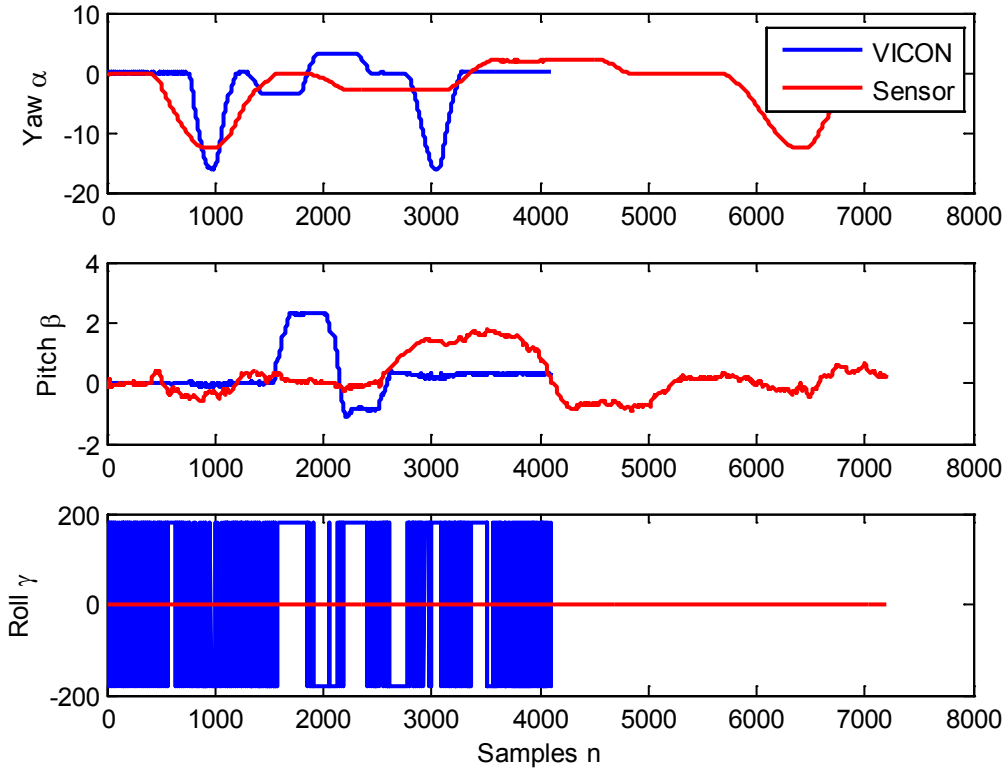


Figure 18. The collected data sets from the sensor and VICON motion system must be synchronized in order to validate the sensor's accuracy.

The data sets from the sensor and VICON motion system were synchronized in two steps. First, the extra measurements at the beginning and end of the VICON's data set were truncated. Second, both of their sampling frequencies were normalized to digital frequencies with natural units. In order to truncate the VICON's data set, an abrupt and distinguishable manipulation of the weapon system was implemented at the beginning and end of each data collection. For example, each trail started and ended with a large yaw to and from the weapon system's reference orientation. These rotations produced distinguishable local maximums or minimums in both data sets that indicated the start and stop of the weapon system's manipulation and allowed for the extra measurements to be ignored. In order to normalize their different sampling frequencies, each data set's digital frequency was computed with

$$\omega = 2\pi \frac{n}{N} \quad (4.1)$$

where ω is its digital frequency in radians per sample, n is the index of an individual sample, and N is the total number of samples in the set. This rationalized the samples' indices in both data sets to an interval of $[0, 2\pi]$. Truncating the VICON's measurements and normalizing to digital frequencies synchronized the collected data. As shown in Figure 19, both data sets start and stop with the yaw's local minimums, and their samples' indices are rationalized to an interval of $[0, 2\pi]$. This allowed for a one-to-one comparison of the orientations that were described by the sensor's estimations and VICON's measurements.

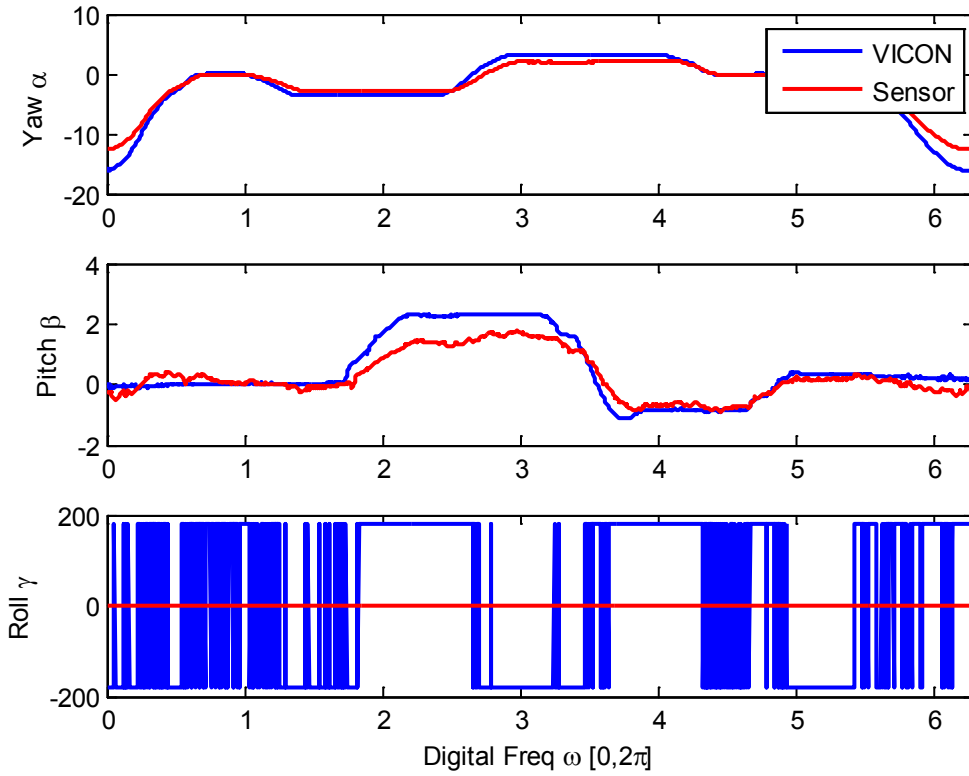


Figure 19. Synchronizing the data sets from the sensor and VICON motion system provided a one-to-one comparison of the orientations that are described by their Euler angles.

2. Accuracy of the Sensor's Estimated Orientations

The desired end state for this experiment is that the sensor's error remained within $\pm 1^\circ$ for dynamic conditions and all orientations. This experiment was conducted in two parts in order to evaluate the sensor's accuracy at various orientations for both static and dynamic conditions. Its error was determined by computing the angular difference between the sensor's estimated orientations and the VICON's ground-truth data. The sensor's accuracy was validated for static conditions, but it did not meet YEI Technology's product specifications for dynamic conditions. The sensor's maximum error that was recorded for both conditions is listed in Table 1.

Table 1. The sensor's accuracy for estimating orientations under static and dynamic conditions.

	Static Conditions	Dynamic Conditions
Sensor Error	$\pm 0.33^\circ$	$\pm 4.07^\circ$

The decrease in accuracy for dynamic conditions was caused by lag from the sensor's onboard filtering processes. As mentioned, it implements Kalman filtering algorithms to estimate orientations from its gyroscope, accelerometer, and magnetic measurements [17]. Its filter update rates are as slow as 200 Hz [15]. This caused the sensor's estimates to lag behind the weapon system's actual orientations as the weapon system was manipulated quickly. Rounding the dynamic error to the next whole number of $\pm 5^\circ$ translates to an offset error of 0.88 meters at a ten-meter range and 43.74 meters at a 500-meter range. At the ten-meter range, two men can stand errantly within an M4 carbine's safety threshold. At a 500-meter range, the distance error is more than half of the safety threshold's radius. Neither of these widths provides sufficient offset from the weapon system's centerline aim point and munitions effects. These standoffs for dynamic conditions are depicted at both ranges in Figure 20. Again, the desired safety threshold for an M4 carbine's cone of fire is depicted with the red circle, and the standoff that remains with the sensor's maximum error is depicted with the blue circle.

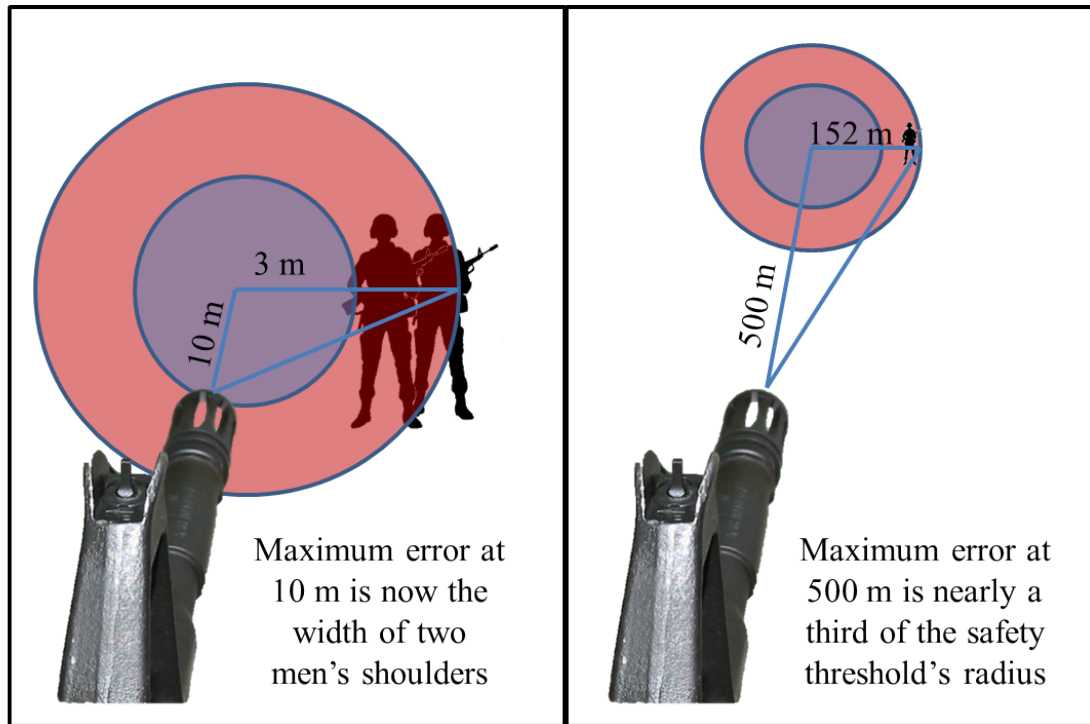


Figure 20. An angular error of $\pm 5^\circ$ causes insufficient standoff at both ranges.
After [8].

The sensor's accuracy met its product specifications when the weapon system was fixed in a static orientation or when it slowly traced the perimeter of the polygon from Figure 16. For these experiments, manipulating the weapon system "slowly" meant that its orientation was changed at a rate of no more than one degree per second. These conditions allowed for the sensor's error to be reduced to its product specifications when the weapon system was being manipulated quicker than this rate. Momentarily fixing the weapon system's orientation or slowing its rate of change provided the sensor's estimates time to overcome the lag from its filter method and to catch up to the weapon system's actual orientations. For example, the weapon system's orientation was momentarily fixed on a point along the polygon's perimeter, or its rate of change was slowed as it traces a corner. During an actual CQC operation, these actions mimic a rifleman acquiring a doctrinal firing position and sighting-in on a target. Riflemen are trained to slow their movement as much as possible in order to provide themselves with enough stability to precisely and accurately engage a target. The reduction in error from these actions is shown in Figure 21. As the weapon system's orientation fixed on a point or its rate of

change slowed to trace a corner, the sensor's error decreased toward its product specifications.

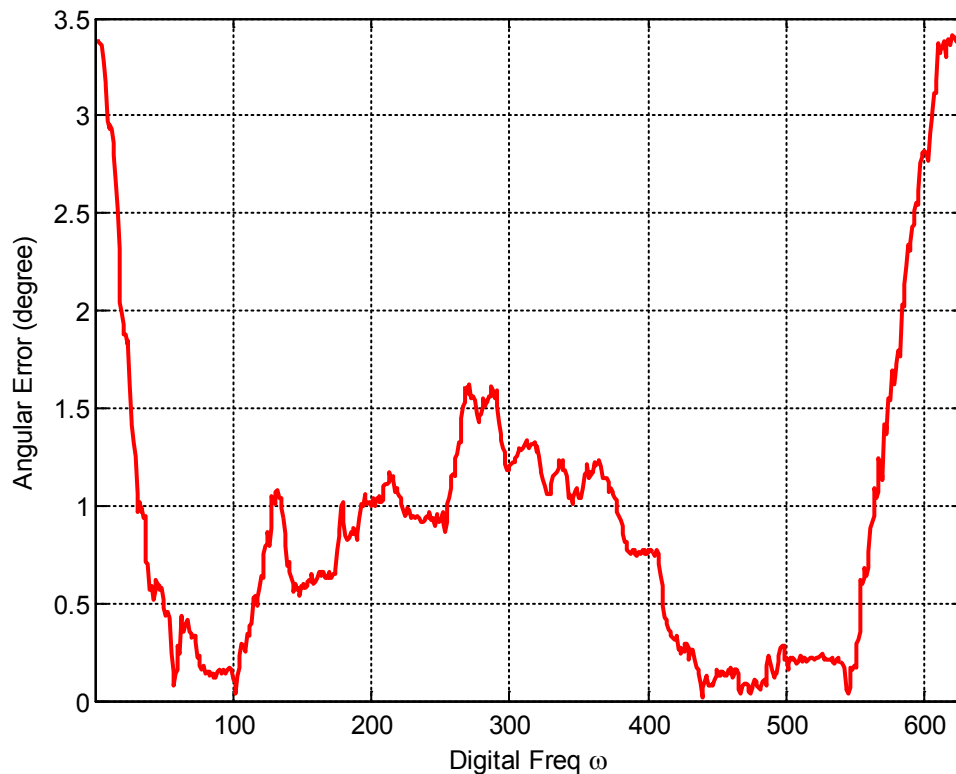


Figure 21. The sensor's error reduces as the weapon system's orientation is fixed or its rate of change is slowed.

Determining the sensor's accuracy for both static and dynamic conditions allowed for progression to the next experiment. The sensor's error for estimating orientations was used to verify the rifleman's kinematic model. Its product specification was validated for static conditions, and it was obtainable for dynamic conditions with actions that mimic how a weapon system is realistically manipulated in a CQC operation. Regardless, the sensor's accuracy for estimating orientations under dynamic conditions was not ideal. As discussed, this IMU was selected without further analysis or comparison to other like devices. A survey of technology is recommended for future work in order to determine the best IMU for this specific application. Filtering this IMU's raw measurements may

also be required in order to optimally reduce the lag produced by any onboard filter methods.

B. VERIFICATION OF THE KINEMATIC MODEL

1. Scaling Factor for the Kinematic Model

The major challenge for verifying the rifleman's kinematic model was projecting the weapon system's orientation vector onto the two-dimensional Cartesian plane. The experiment for validating the sensor's accuracy was repeated, but the objective for this iteration was to evaluate the model's description of the weapon system's motion. The PEQ-15's visible laser was selected as the feedback mechanism for this experiment since it is used as an aim point in CQC. In order to evaluate the model's trueness, its end point on the Cartesian plane was compared to the polygon's perimeter in Figure 16. Ideally, the end point of this laser should trace the polygon's perimeter within the sensor's level of accuracy. A scaling factor was required to adjust the length of the laser's vector as the weapon system was manipulated. This ensured that the vector's endpoint laid in the Cartesian plane in order to verify that it traced the polygon's perimeter as desired.

The scaling factor helped to obtain the grid coordinates that corresponded to the laser's projection onto the two-dimensional Cartesian plane. Even though this is not required for an actual CQC operation, determining these grid coordinates was needed to map the end of the laser's beam that physically traced the perimeter from Figure 16. The length of the laser's vector started at the weapon system's range to the plane with the weapon system in its reference orientation. If the vector's length remained constant, its end point would not have laid in the plane while the weapon system was manipulated. Unless the weapon system was in its reference orientation, the vector's length was too short. As such, its length was extended proportionally to the weapon system's angle from its reference orientation. As shown in Figure 22, the length of the vector needed to be adjusted precisely. Extending its length too little or too much added error when analyzing the model's trueness. Their endpoints needed to be projected orthogonally onto the plane. Since the reference orientation was the only orientation that is orthogonal to the plane, the remaining projections added error for a vector of errant length.

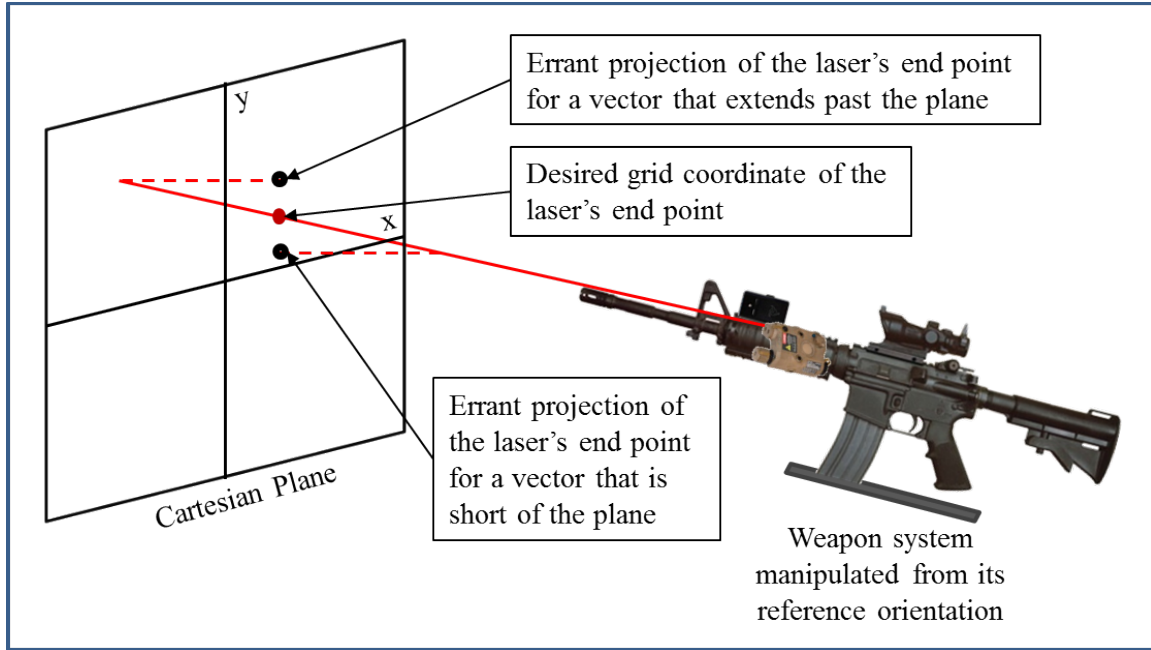


Figure 22. A scaling factor adjusted the length of the laser's vector to ensure that its endpoint remained in the two-dimensional Cartesian plane as the weapon system was manipulated. After [12]–[15].

Other means for determining the laser's grid coordinates were investigated, but computing this scaling factor was selected since it was the quickest and least computationally expensive method that was explored. For example, another means for determining the grid coordinates on the plane is to extend the length of the laser's vector to the weapon system's maximum effective range. A 500-meter length ensures that the vector always intersected the plane since this experiment was conducted at a range of ten meters or less. Calculus could have been used to compute the intersecting grid coordinate between the vector and plane. This method was not selected due to the computational cost of determining their intersection. The scaling factor l can be computed with the single equation

$$l = \frac{{}^y x_p \cos(\alpha) + {}^p z_L \cos(\alpha) \sin(\beta) + r}{\cos(\alpha) \cos(\beta)} \quad (4.2)$$

where r is the weapon system's range to the plane and the remaining variables were defined in Chapter II. The scaling factor l adjusted the length of $\vec{W}$'s x -component by

redefining the vector as $\vec{W} = (l, 0, 0, 1)^T$. The one was concatenated to the bottom of $\vec{W}$ in order to match its dimensions with the transformation operators from Chapter II. After it was resized, the vector $\vec{W}$ was multiplied with ${}^{NED}_L T$ to describe the laser's endpoint as it traced the polygon's perimeter.

2. Trueness of the Kinematic Model

The desired end state of this experiment was that the rifleman's kinematic model properly described the weapon system's motion. The PEQ-15's visible laser was used to trace the polygon from Figure 16, and the model was responsible for reproducing the movement of the laser's endpoint on the Cartesian plane. The model described this motion with the data sets that were collected from the sensor and VICON motion system. Human error was inherent in this experiment since the weapon system was manipulated. As discussed in Chapter III, the VICON's ground-truth data helped to evaluate if this additional error detrimentally affected the model's description. Since the model described the laser motion as desired with the VICON's data set, the model of a rifleman's kinematics was verified despite the human error from tracing the perimeter and setting up the plane. It was still important to evaluate the model's description of the laser's motion with the sensor's data set. This confirmed that the sensor's accuracy remained the same when integrated into the kinematic model. Its estimations were compared to the VICON's ground-truth data, and the model's description of the laser's motion is shown on the Cartesian plane in Figure 23. The weapon system started in its reference orientation with the laser pointing at the plane's origin. After a distinguishable yaw to synchronize the collected data sets, the laser traced the polygon's perimeter starting to the left.

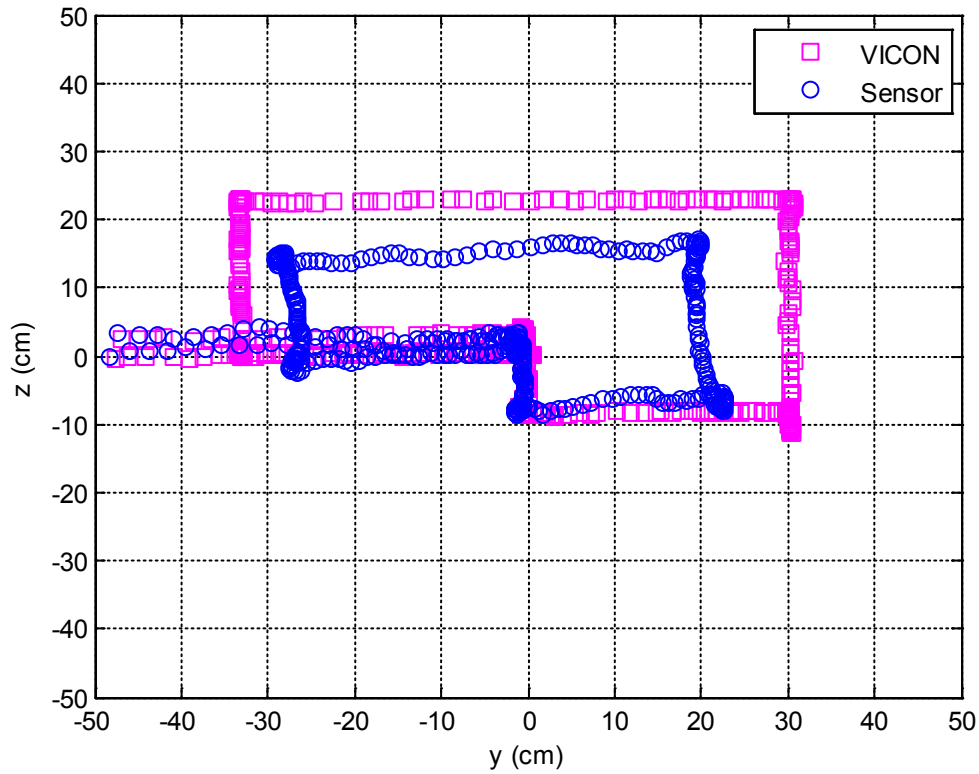


Figure 23. The model's description of the weapon system's orientation as its PEQ-15 visible laser traces the polygon's perimeter.

The sensor's accuracy was maintained when integrated with the kinematic model. This experiment was repeated for multiple trials within a 4–7 meter range from the weapon system to the plane. This range was bounded by available space in the laboratory. It is important to note that the sensor's estimates consistently under estimated the rotations of the weapon system. As shown in Figure 23, the polygon that the sensor traced remained compressed relative to the VICON's polygon. Two potential causes were investigated. First, the scaling factor l may have been computed or implemented incorrectly. Second, the sensor's calibration needed to be confirmed. Neither of these appeared to be the cause of the sensor's compression. The model accurately described the laser as is traced the polygon with the VICON's ground-truth data. Since the scaling factor was computed and implemented the same way for the sensor's data, it was concluded that the scaling factor was not adjusting the vector's length to be too short or

too long. Additionally, the grid coordinates for the laser's endpoints were computed with three-dimensions. The x and y components were the endpoint locations on the plane, and its z component was used to verify that this location was in the plane. Since this z component remained at zero, it was again concluded that l was working as desired and the endpoint lied in the plane. The sensor's calibration and settings were confirmed, and the experiment was repeated multiple times with multiple sensors. The sensors consistently under estimated the rotations within the desired level of accuracy.

The VICON's ground-truth data validated the model's trueness, but it was important to see how the sensor's error would be impacted as the system's prototype is incrementally constructed. The sensors' under estimation of the weapon system's rotations should be investigated further in future work. If the YEI 3-Space Data-Logging sensor is selected as the best IMU for this specific application, a least-squares fit or polynomial interpolation is recommended to account for this compression. The model's description of the weapon system's orientation must be as accurate as possible to avoid false positive and negative detections from the geometry-of-fire tracking algorithm. Despite the compression consideration, the transformation operators that were developed for this kinematic model were verified. The tracking algorithm could now be tested with the sensor's accuracy validated and the kinematic model verified.

C. TESTING OF THE TRACKING ALGORITHM

1. Simulation of Moving Friendly Nodes

Testing the tracking algorithm did not present a major challenge that adjusted its experimental design. After integrating the tracking algorithm into the network, the designed experiment evaluated the algorithm as desired for the four scenarios; however, the major lesson learned from this experiment was that the VICON motion system could have been incorporated to provide positional data. The original experimental design did not call for the VICON motion system since it had been used to provide ground truth orientation data up to this point. Since the experiments built off each other, the sensor's accuracy was confirmed and the model's trueness was verified, it did not seem necessary to incorporate the VICON motion system to track orientations anymore. It could have

been used to track the positions of the friendly nodes for the network. These positions could have been manipulated in real time instead of having to simulate their movements for each scenario. Integrating the VICON motion system would have allowed for more realistic testing and further analysis of the algorithm and network. The algorithm's false positive and negative detections could have been evaluated, and uncertainties and confidence intervals could have been explored. The realization for incorporating the VICON motion system happened at the end of this thesis research, and the student who developed the tactical network graduated from NPS before it could be integrated. As such, this integration is recommended for future work.

2. Algorithm's Ability to Detect Geometry Issues

The tracking algorithm was tested for the four scenarios from the experimental design. The algorithm properly identified conflicted geometries as desired. Since a human study is required to determine the best feedback mechanism, the rifleman was alerted of geometries issues with a message displayed on a computer screen. An example of this output is shown in Figure 24. The message simply warned the rifleman to cease fire and informed them of the angle to a friendly node.



```
*****  
CEASE FIRE: Geometries must be de-conflicted!  
Angle: 16.9  
*****
```

Figure 24. An example of the *friendly detection* script's output that alerts a rifleman of a present geometry issue.

The major questions that stirred from these scenarios were: should the algorithm's loop be time or event-based, how fast does the algorithm need to update, and how are false positives and negatives analyzed? Since operations are event based, it was decided

that the algorithm should also be event-based. The algorithm should compute geometries as the weapon system's orientations and friendly nodes' positions change. This allowed the algorithm to update as need, but the problem was ensuring that it updated fast enough. A black-box analysis was conducted to determine this rate. The three feedback mechanisms that were discussed could be a light flashing in the ACOG, a tone beeping in an earpiece, or a vibration in the weapon system's butt stock. Since light flashing in the optic is the quickest, it was used to evaluate if the event-based algorithm operated fast enough. The eye is able to perceive a blinking light at up to 20 Hz [20]. As such, this rate was used for the black-box analysis. The number of nodes was increased from a fire team level to a squad level in order to evaluate how quickly the algorithm could compute and identify conflicted geometries. Even with the simulated movement of 13 friendly nodes, the algorithm was still able to update positions and orientations and to detect geometry issues at the desired 20 Hz rate. The scenarios that involved moving nodes brought up the discussion about false positives and negatives. As discussed, integrating the VICON motion system to provide positional data should allow for this additional analysis.

D. SUMMARY

All three of the experiments presented major challenges that adjusted the original test plans. The data sets that were collected to validate the sensor's accuracy needed to be synchronized. A scaling factor was needed to adjust the length of the laser's orientation vector in order to verify that the kinematic model properly described the weapon system's motion. Additionally, simulating the moving positions for friendly nodes was required to test the tracking algorithm. All three of the experiments also had considerations with their results. For dynamic conditions, the sensor's accuracy only met its product specifications when the weapon system's orientation was momentarily fixed or slowed its rate of change. The sensor's estimations compressed the model's description of the weapon system's manipulations. Furthermore, the tracking algorithm highlighted the false negative and false positive issue that needs to be addressed with the uncertainties from the position and orientation data. Ultimately, these considerations directly apply to recommendations for future work. These recommendations and the original contribution of this research are discussed in the final chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION AND RECOMMENDATIONS

A. SUMMARY OF CONTRIBUTIONS

The goal of this research was to develop a system that streamlines the geometry-of-fire validation cycle for direct-fire weapon systems. This technology will provide dismounted infantry units with a tactical advantage by enhancing their situational awareness and increasing their tempo. As discussed in Chapter I, this system was designed as a tactical network consisting of five subsystems. These subsystems correspond to the following components: transmitters/receivers that communicate with the network, sensors that track riflemen's positions, sensors that track weapon systems' orientations, algorithms and software that compute geometries between friendly nodes, and feedback mechanisms that inform riflemen of conflicted geometries. Two of these subsystems were developed, with the focus on tracking a weapon system's orientation and computing the offsets between friendly nodes in the network. The desired end state was to validate these subsystems' designs in order to integrate them into a prototype of the tactical network.

A kinematic model of a rifleman and geometry-of-fire tracking algorithm were developed for these subsystems. The kinematic model uses an orientation estimation sensor to describe the motion of a manipulated weapon system, and the tracking algorithm identifies conflicted geometry of fires when integrated with a tactical network. As detailed in Chapter IV, both of these subsystems were tested and verified. Their current designs are not ready to be integrated into the system's final design, but they serve as proofs-of-concept and provide viable foundations for future refinement. The design requirements for this thesis were simplified since this was the first attempt to develop these subsystems. Future iterations need to address these simplifications. The rifleman's kinematic model must be expanded to describe an actual human body, and a survey of technology is recommended to identify the best IMU for this specific application. Once a PNS is designed, the tracking algorithm also needs further evaluation. It was only tested for the four scenarios that were detailed in Chapter III, and the movements of node positions were simulated during the testing of these scenarios. With a

PNS to track riflemen's positions, the algorithm's false positives and negatives for detection can be analyzed and uncertainties and confidence intervals can be computed.

The major contribution of this thesis is the exploration of a novel idea. A tactical network has never been designed for this specific application. The idea to provide dismounted infantry units with this technological advantage came from the students and faculty of the Department of Electrical and Computer Engineering at NPS. Merging the students' military experience with the faculties' academic knowledge was instrumental in the design of the overall system and development of these subsystems. The results from this thesis demonstrate that a weapon system's orientation can be tracked and the offsets between friendly nodes can be computed. As the kinematic model and tracking algorithm are refined in future work, these subsystems can be extended to other direct-fire weapon systems, and the design of the overall system can be expanded to other types of operations and environments. This system will ideally replace the antiquated *300 Mil Rule*. If all other things are equal, this technology could provide riflemen with the edge that is needed to tip the scales of war in their favor.

B. FUTURE WORK

(1) Survey of Technology

A survey of technology was not completed before selecting the YEI 3-Space Data-Logging sensor for this system's design. This survey is recommended in order to determine the best IMU for this specific application. As discussed, the YEI 3-Space Data-Logging sensor was selected since it is readily available and boasts the desired level of accuracy for orientation estimation. This level of accuracy was verified for static orientations but diminished as the weapon system's orientation changed quickly under dynamic conditions. Riflemen must be able to move quickly. The main cause for this drop in accuracy is due to lag from the sensor's filter method. This sensor is marketed for a wide-range of applications. Another IMU may be designed, or at least better suited, for this specific application.

(2) Filtering of the Sensor's Raw Measurements

Filtering of the sensor's raw measurements should also be investigated in order to identify the best filter design for this specific application. The YEI 3-Space Data-Logging sensor used Kalman filtering techniques to process its measurements. As discussed, this sensor was designed for a wide-range of applications, and the lag between the IMU's measurements and its filter's estimates limited its accuracy for dynamic conditions. Other filtering techniques can potentially reduce this lag and help maintain the sensor's level of accuracy for this specific application [3]. This recommendation for future work should be completed after the survey of technology as the filter's design depends on the IMU that is selected.

(3) Refinement of the Rifleman's Kinematic Model

The model of the rifleman's kinematics was simplified for this thesis. A rifleman carrying a direct-fire weapon system was represented by an M4 carbine mounted to a tripod. The tripod reduced the number of degrees of freedom and allowed the M4 carbine to be held in fixed positions and orientations. Even though this helped verify the model and analyze the sensor's accuracy, the rifleman's kinematic model needs to be refined for an actual human body. This is imperative for tracking a weapon system's rotations and translations as it is manipulated during an actual CQC operation. Open-source kinematic models of the human body have been developed for other applications. These must be investigated in order to determine the attributes of the human body that should be taken into account for this specific application.

(4) Integration of the PNS, Tactical Network, and Feedback Mechanism

Prototypes of the system's remaining components must be designed, validated, and integrated. Two of the system's five functions were addressed in this thesis: tracking the weapon system's orientation and computing the geometry of fires at each friendly node. As discussed in Chapter II, concurrent thesis research at NPS is the design of a tactical network, but the PNS and feedback mechanisms still need to be explored. Designing the PNS is arguably the most challenging. Its level of accuracy must be as high as possible, and it must be maintained for the duration of a CQC operation. Designing the

feedback mechanism requires human subject research. This mechanism could be a light flashing in the ACOG, a tone beeping in an earpiece, or a vibration in the weapon system's butt stock. A large population of experienced riflemen should evaluate these mechanisms in order to determine which works best for this specific application. All of these components need to be integrated into a single prototype of the complete system once they have been individually designed and validated.

APPENDIX

A. SETTINGS FOR THE YEI 3-SPACE DATA-LOGGING SENSOR

The default manufacturer settings for the YEI 3-Space Data-Logging sensor were adjusted for this thesis' specific application. Its default coordinate system was changed from a left-handed system to a right-handed system. In order to align its coordinate directions with the NED reference frame, the sensor's x -axis pointed out of its front, y -axis pointed out of its right-side, and z -axis pointed straight down. This configuration is depicted in Figure 25.

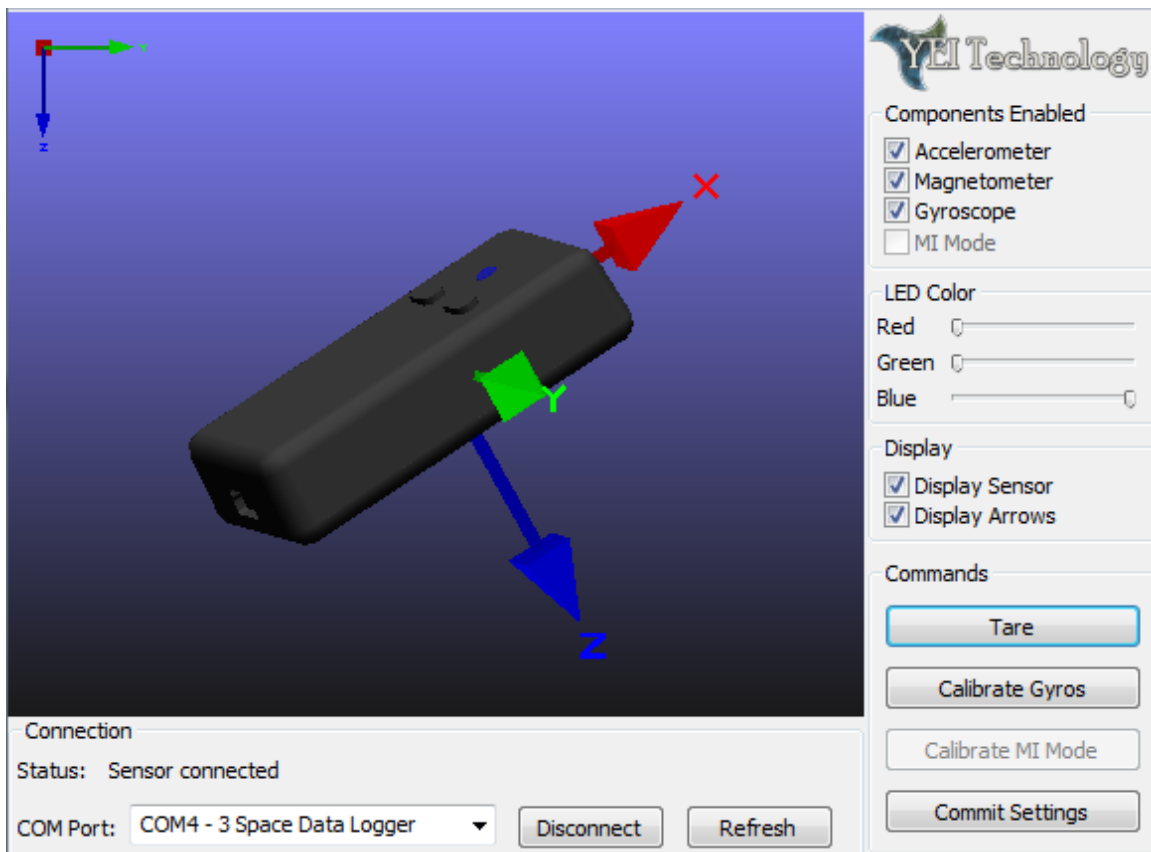


Figure 25. The default coordinate system was adjusted to align its coordinate directions with the NED reference frame as depicted. From [21].

Additionally, the sensor's default advanced settings were adjusted. Its filter mode was changed from *Kalman* to *Q-Comp*. The *Q-Comp* filter mode stands for quaternion

complementary. According to the sensor's user manual, this filter is a faster method for estimating orientation with an accuracy that is comparable to its *Kalman* filter mode [21]. The sensor's running average was set to 15.000 in order to balance the smoothness of its estimates with the lag from its filter method. The sensor's reference mode was set to *Single Auto Continual* in order to mitigate the effects of shifts in magnetic force and instabilities of the sensor's accelerometer [21]. The changes to these settings are shown in Figure 26.

The screenshot shows a configuration window for a sensor. It is divided into several sections:

- ometer Trust Values:** Contains radio buttons for 'Static' and 'Confidence'. 'Confidence' is selected, with a range from 0.001000000475 to 0.0625.
- Magnetometer Trust Values:** Similar to the first section, with 'Confidence' selected and the same range.
- Axis Directions:** A dropdown menu shows '(X: Forward, Y: Right, Z: Down) (Right-handed)'. Below it are checkboxes for 'Negative X', 'Negative Y', and 'Negative Z', with 'Negative Z' checked.
- Misc.:** Contains 'Filter Mode' set to 'Q-Comp', 'Oversample Rate' set to 1, and 'Running Average' set to 15.000.
- Reference Modes:** A table with columns 'Accel' and 'Compass'. The 'Single Auto Continual' mode is selected.

Red arrows highlight the 'Filter Mode' (Q-Comp), 'Running Average' (15.000), and 'Single Auto Continual' reference mode.

	Accel	Compass
Single Manual	0.0	0.514459073544
Single Auto	0.0	0.0
Single Auto Continual	-1.0	0.85751491785

Figure 26. The sensor's filter mode, running average, and reference mode were adjusted as shown. After [21].

B. MATLAB SCRIPTS FOR VERIFYING THE KINEMATIC MODEL

MATLAB programming language and computing environment was used to verify the model of a rifleman's kinematics. Two functions and a script file were written to conduct batch mode processing of the data sets that were collected from the sensor and VICON motion system. The first function was called by the script file to compute the sensor's error, and the second function was called by the script file to synchronize the data sets for plotting purposes. The script file served as a main program. It called the two functions and implemented the kinematic model.

1. Function for Analyzing the Sensor's Accuracy

```
function [Error,pA,pM,kAstart,kAstop,kMstart,kMstop] =  
sensorerror(pActual,pMeasured,laserActual,laserMeasured)  
  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
% Compute and display the distance and angular error between the  
% actual and measured grid coordinates. The displayed error  
% represents the error in the sensor's readings in comparison to  
% the ground-truth data collected from the VICON system.  
% INPUTS:  
%   pActual -- Ground truth grid coordinates from the VICON system  
%   pMeasured -- Measured grid coordinates from the sensor's  
%               readings  
%   laserActual -- Laser position in the grid's coordinate system  
%               from the VICON system (needed for the angular  
%               error)  
%   laserMeasured -- Laser position in the grid's coordinate system  
%               from the sensor (needed for the angular error)  
%   (assume that the number of laser positions equals the number of  
%   grid coordinates from their respective systems)  
% OUTPUTS:  
%   Displays the mean, minimum and maximum distance and angular  
%   errors  
%   Error -- Distance error in the first row and angular error in  
%           the second row  
%   pA -- Truncated set of actual grid coordinates from the VICON  
%         system  
%   pM -- Truncated set of measured grid coordinates from the  
%         sensor  
%   kAstart -- Synchronization index to start the VICON data set  
%   kAstop -- Synchronization index to stop the VICON data set  
%   kMstart -- Synchronization index to start the sensor's data set  
%   kMstop -- Synchronization index to stop the sensor's data set  
  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
% The VICON system and sensor collect data at different sampling
```

```

% rates. In order to compare the sensor's readings to the correct
% VICON grid coordinates, we must ensure that all matrices
% (pActual, laserActual, pMeasured and laserMeasured) are the same
% size. These matrices should have the same number of rows (ie.
% two for the grid coordinates and three for the laser positions)
% but varying number of columns (ie. difference between the VICON
% system's and sensor's sampling rates).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine when the sensor starts recording motion in the rifle.
% This will allow us to ignore the initial readings from the sensor
% when the rifle is still at its reference position

% Determine the maximum distance from the origin in the first half
% of the data
dist=[];
for i=1:floor(length(pMeasured(1,:))/2)
    dist=[dist sqrt(pMeasured(:,i)'*pMeasured(:,i))];
end
k=find(dist==max(abs(dist)));
k=k(1);
kMStart=k;

% Remove the initial sensor readings while the rifle is still in
% the reference position
pMeasured=pMeasured(:,k:end);
laserMeasured=laserMeasured(:,k:end);

% Repeat the same steps to determine when the VICON system starts
% recording motion in the rifle.

% Determine the maximum distance from the origin in the first half
% of the data
dist=[];
for i=1:floor(length(pActual(1,:))/2)
    dist=[dist sqrt(pActual(:,i)'*pActual(:,i))];
end
k=find(dist==max(abs(dist)));
k=k(1);
kAStart=k;

% Remove the initial VICON readings while the rifle is still in the
% reference position
pActual=pActual(:,k:end);
laserActual=laserActual(:,k:end);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine when the sensor stops recording motion in the rifle.
% This will allow us to ignore the final readings from the sensor
% when the rifle returns to its reference position

% Determine the maximum distance from the origin in the second half

```



```

% of the data
dist=[];
for i=round(length(pMeasured(1,:))/2):length(pMeasured(1,:))
    dist=[dist sqrt(pMeasured(:,i)'*pMeasured(:,i))];
end
k=find(dist==max(abs(dist)));
k=floor(length(pMeasured(1,:))/2)+k(end);
KMStop=k;

% Remove the final sensor readings when the rifle returns to the
% reference position
pMeasured=pMeasured(:,1:k);
laserMeasured=laserMeasured(:,1:k);

% Repeat the same steps to determine when the VICON system stops
% recording motion in the rifle.

% Determine the maximum distance from the origin in the second half
% of the data
dist=[];
for i=round(length(pActual(1,:))/2):length(pActual(1,:))
    dist=[dist sqrt(pActual(:,i)'*pActual(:,i))];
end
k=find(dist==max(abs(dist)));
k=floor(length(pActual(1,:))/2)+k(end);
kAStop=k;

% Remove the initial VICON readings while the rifle is still in the
% reference position
pActual=pActual(:,1:k);
laserActual=laserActual(:,1:k);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine which matrix contains more grid coordinates
% ('a' stands for actual and 'm' stands for measured)
% (the number of laser positions should equals the number of grid
% coordinates from their respective systems)
dimA=size(pActual);
dimM=size(pMeasured);

% Construct the digital frequency vectors for each data set
kA=1:dimA(2);
kM=1:dimM(2);
wA=2*pi*kA/dimA(2);
wM=2*pi*kM/dimM(2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute and display the mean, minimum and maximum distance errors
% between the actual and measured grid coordinates

% Compute the difference between the actual and measured grid
% coordinates

```

```

dist=[];
pA=[];
pM=[];
laserA=[];
laserM=[];
for w=0.01:0.01:2*pi

    % Determine the integer index that corresponds to the current
    % digital frequency w
    kAIndex=round(w*dimA(2)/(2*pi));
    kMIndex=round(w*dimM(2)/(2*pi));

    % Append the computed distance between the two sets of data
    dist=[dist pActual(:,kAIndex)-pMeasured(:,kMIndex)];

    % Append the indexed data sets that are being compared for the
    % angular error calculations
    pA=[pA pActual(:,kAIndex)];
    pM=[pM pMeasured(:,kMIndex)];
    laserA=[laserA laserActual(:,kAIndex)];
    laserM=[laserM laserMeasured(:,kMIndex)];
end

% Loop through the number of grid coordinates
dim=size(dist);
distError=[];
for k=1:dim(2)

    % Compute the distance error for each grid coordinate from the
    % calculated difference
    distError=[distError sqrt(dist(:,k)' $\cdot$ dist(:,k))];

end

% Display the mean, minimum and maximum distance errors
disp('The distance errors are (in centimeters): ')
distError_Mean=mean(distError)
distError_Mode=mode(distError)
distError_Median=median(distError)
distError_Min=min(distError)
distError_Max=max(distError)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute and display the mean, minimum and maximum angular errors
% between the actual and measured grid coordinates

% Create a row of zeros to append to the grid coordinate matrices
zero=[1:dim(2)];
zero=zeros*0;

% Append this row of zeros to the top of the grid coordinate
% matrices to serve as the x-component of these vectors (remember

```

```

% that these grid coordinates are in the grid's (y,z) plane)
pA=[zero; pA];
pM=[zero; pM];

% Compute the position vectors that represent the laser's beam.
% This vector will be in the grid's system from the laser's
% position to the grid coordinates in the (y,z) plane.
pA_LG=pA-laserA;
pM_LG=pM-laserM;

% Loop through the number of position vectors
angleError=[];
for k=1:dim(2)

    % Compute the angular error for each grid coordinate
    angleError=[angleError atan2(norm(cross(pA_LG(:,k),pM_LG(:,k))),...
                                   dot(pA_LG(:,k),pM_LG(:,k))))];

end

% Display the mean, minimum and maximum angular errors
disp('The angular errors are (in degrees): ')
angleError_Mean=mean(angleError)*180/pi
angleError_Mode=mode(angleError)*180/pi
angleError_Median=median(angleError)*180/pi
angleError_Min=min(angleError)*180/pi
angleError_Max=max(angleError)*180/pi

% Create the Error variable for the output
Error=[distError;
       angleError*180/pi];

end

```

2. Function for Synchronizing the Collected Data Sets

```

function [AVSampled,ASSampled,wVSampled,wSSampled] =
samplerate(AV,AS,kASStart,kASStop,kMStart,kMStop)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The VICON system and sensor collect data at different sampling
% rates. In order to compare the sensor's readings to the correct
% VICON grid coordinates, we must ensure that all vectors of angles
% (Az, Ay & Ax) are the same size. These matrices should have the
% one row but varying number of columns (ie. difference between
% the VICON system's and sensor's sampling rates).
% INPUTS:
%   AV -- VICON angular readings in radians
%   AS -- Sensor angular readings in radians
% OUTPUTS:
%   AVSampled -- Truncated VICON angular readings in radians

```

```

% ASSampled -- Truncated sensor angular readings in radians
% wVSampled -- Digital frequency for the VICON readings
% wSSampled -- Digital frequency for the sensor readings
% kAstart -- Synchronization index to start the VICON data set
% kAstop -- Synchronization index to stop the VICON data set
% kMstart -- Synchronization index to start the sensor's data set
% kMstop -- Synchronization index to stop the sensor's data set

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Truncate the data sets with the synchronization indices
AS=AS(kMstart:end);
AS=AS(1:kMstop);
AV=AV(kAstart:end);
AV=AV(1:kAstop);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine which vector contains more angles
dimA=length(AV); % Actual dimension from the VICON system
dimM=length(AS); % Measured dimension from the sensor

% Construct the digital frequency vectors for each data set
kA=1:dimA;
kM=1:dimM;
wA=2*pi*kA/dimA;
wM=2*pi*kM/dimM;

% Define the output variables
AVSampled=AV;
ASSampled=AS;
wVSampled=wA;
wSSampled=wM;

end

```

3. Main Script File for Implementing the Kinematic Model

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% THESIS Sensor & VICON Integration %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clear variables
clear all
clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Model the coordinate systems in relation to a reference frame
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% List of the coordinate systems
% {0}: NED

```

```

% {T}: Tripod
% {R}: Rifle
% {L}: Laser
% {G}: Grid
% {S}: Sensor
% {V}: VICON
% {A}: Actual NED

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define the constant variables that describe the distances between the
% different coordinate system origins
h_TO=102; % Distance between the NED's and tripod's origins in centimeters
          % along the negative z-axis of the NED's system
          % (height of the tripod's origin from the floor)
h_LR=17; % Distance between the rifle's and laser's origins in centimeters
          % along the negative z-axis of the rifle's system
          % (height of the laser's origin from the rifle)
h_SL=4; % Distance between the laser's and sensor's origins in centimeters
          % along the positive z-axis of the laser's system
          % (height of the sensor's origin from the laser)
h_RT=7; % Distance between the tripod's and rifle's origins in centimeters
          % along the negative z-axis of the tripod's system
          % (height of the rifle's origin from the tripod)
h_GO=h_TO+h_RT+h_LR; % Distance between the NED's and grids's origins in
          % centimeters along the negative z-axis of the NED's system
          % (height of the grid's origin from the floor)
d_GO=560; % Distance between the NED's and grid's origins in centimeters
          % along the negative x-axis of the NED's system
          % (distance from the grid's origin to the NED)
          % d_GO=700 cm for the far position
          % d_GO=420 cm for the near position
          % d_GO=560 cm for the networked position
d_RT=2.5; % Distance between the tripod's and rifle's origins in centimeters
          % along the negative x-axis of the tripod's system
          % (offset of the rifle's origin from the tripods)
d_SL=42; % Distance between the laser's and sensor's origins in centimeters
          % along the negative x-axis of the laser's system
          % (offset of the sensor's origin from the laser)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define the transformation matrices between the coordinate systems
% that are fixed to each other with the reference angles
% Az: rotation about the z-axis (yaw)(alpha)
% Ay: rotation about the y-axis (pitch)(beta)
% Ax: rotation about the x-axis (roll)(gamma)
% Angles must be in radians and set to zero for the initial reference
% position (rifle level and perpendicular to the grid while facing it)
Az=0*pi/180;
Ay=0*pi/180;
Ax=0*pi/180;

% Remember: O_BA is the {B}'s origin position relative to {A}
T_GO=[ 1  0  0 -d_GO; % Describe {G} relative to {O}

```

```

0 -1 0 0;
0 0 -1 -h_GO;
0 0 0 1];

T_TO=[-cos(Az) sin(Az) 0 0; % Describe {T} relative to {O}
-sin(Az) -cos(Az) 0 0;
0 0 1 -h_TO;
0 0 0 1];

T_RT=[ cos(Ay) 0 sin(Ay) -d_RT; % Describe {R} relative to {T}
0 1 0 0;
-sin(Ay) 0 cos(Ay) -h_RT;
0 0 0 1];

T_LR=[ 1 0 0 0; % Describe {L} relative to {R}
0 1 0 0;
0 0 1 -h_LR;
0 0 0 1];

T_SL=[ 1 0 0 -d_SL; % Describe {S} relative to {L}
0 1 0 0;
0 0 1 h_SL;
0 0 0 1];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define the compound transformations between the coordinate systems
% that are not directly fixed to each other.
T_LG=inv(T_GO)*T_TO*T_RT*T_LR; % Describe {L} relative to {G}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define the position vector in the laser's coordinate system that ends
% at the grid's origin when all of the systems are in their initial ref
% positions (ie. rifle must be level and perpendicular to the grid
% while facing it).

% Determine the scaling constant for the vector to end in the grid's
% (y,z) plane
c=(d_RT*cos(Az)+h_LR*cos(Az)*sin(Ay)+d_GO)/(cos(Ay)*cos(Az));

% Scale the x-component of the laser's position vector since it is
% aligned with the system's x-axis
pr_L=[c;
0;
0;
1];

% Describe this position vector in the grid's coordinate system after
% it has been scaled. Ensure that the x-component of the grid's pos
% vector is zero.
pr_G=T_LG*pr_L;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% Determine the scaling factor the rotations about the y-axis
diff=[1.36 1.56 1.83; % actual reading divided by measured reading
      250 420 700]; % distance from {G} to {O} in centimeters

% Determine the fitted polynomial for these differences
P=polyfit(diff(2,:),diff(1,:),2);

% Compute the scaling factor y for the distance from {G} to {O}
y=P(1)*d_G0^2+P(2)*d_G0+P(3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine the measured grid coordinates that track the rifle's
% orientation with the sensor's readings
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% We must relate the actual NED system to our initial reference
% position!
% 1. In order to relate our drawn systems to the actual NED system, we
% can take an initial sensor reading with the rifle level and
% perpendicular to the grid while facing it (ie. sensor aligned with
% our drawn system in the reference position). This will provide us
% with the offset between the drawn and actual systems.

% Import the sensor data (include the .txt extension with the file
% name)
Q_SiA=dataReader('dataSensorBoxNetworkRef.txt');

% Average these quaternions
q_SiA=quatavg(Q_SiA);

% Convert these quaternions into rotation matrix that describe the
% offset to the actual systems in the initial reference position
R_SiA=quat2dcm(q_SiA); % Describe {S} relative to {A}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Import the quaternions from the sensor's readings at various angles
% of rotation
% Include the .txt extension with the file name
Q_SkA=dataReader('dataSensorBoxNetwork.txt');

% Convert these quaternions into rotation matrices
R_SkA=quat2dcm(Q_SkA);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine the measured angles of rotation from the sensor's initial
% rotation matrix R_SiA in the reference position

% Loop through the number of quaternions
dim=size(R_SkA);
p_G=[];
laserMeasured=[];
Azs=[];

```

```

Ays=[];
Axs=[];
for k=1:dim(3)

    % Describe the angles of rotation from sensor reading k relative to
    % the initial sensor reading i
    R_SkSi=inv(R_SiA)*R_SkA(:, :, k); % Describes {Sk} relative to {Si}

    % Determine the measured angles of rotation from this matrix
    % Azm: rotation about the z-axis (yaw)(alpha)
    % Aym: rotation about the y-axis (pitch)(beta)
    % Axm: rotation about the x-axis (roll)(gamma)
    % Angles must be in radians ('m' stands for measured)
    [Azm Aym Axm]=dcm2angle(R_SkSi);

    % Rescale the rotations about the y-axis with the computed scaling
    % factor y
    % Aym=abs(y)*Aym;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Redefine the transformation matrices with the measured angles of
    % rotation from the sensor's readings. These rotations are about
    % fixed axis and should be pre-multiplied by their respective
    % coordinate systems
    % (ie. rotations about the z-axis should be pre-multiplied with the
    % tripod's system and rotations about the y-axis should be pre-mult
    % with the rifle's system).

    % Define the special homogeneous transformations for the rotations
    % about a fixed axis
    Tz=[ cos(Azm) -sin(Azm) 0 0;
         sin(Azm)  cos(Azm) 0 0;
         0         0        1 0;
         0         0        0 1];
    Ty=[ cos(Aym) 0 sin(Aym) 0;
         0      1 0      0;
        -sin(Aym) 0 cos(Aym) 0;
         0      0 0      1];
    Tx=[ 1      0      0 0;
         0 cos(Axm) -sin(Axm) 0;
         0 sin(Axm)  cos(Axm) 0;
         0      0      0 1];

    % Redefine the description of {L} relative to {G} by pre-multiply
    % rotations by their respective coordinate systems. This allows us
    % to determine the resulting position vectors from these rotations
    % ('m' stands for measured)
    T_LGm=inv(T_GO)*T_TO*Tz*T_RT*Ty*Tx*T_LR;

    % Retrieve the Euler angles from R_LGm in order to plot and
    % compare to the VICON system angles
    R_LGm=T_LGm(1:3,1:3)*[cos(pi) 0 sin(pi); 0 1 0;-sin(pi) 0 cos(pi)];

```



```

[Azr Ayr Axr]=dcm2angle(R_LGm);
Azs=[Azs Azr];
Ays=[Ays Ayr];
Axs=[Axs Axr];

% Collect the laser positions in the grid's coordinate system in
% order to calculate the angular error between the sensor's
% readings and the VICON system
laserMeasured=[laserMeasured T_LGm(1:3,4)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute the resulting position vectors from these rotations in
% the grid's coordinate system. These vectors should end in the
% grid's (y,z) plane
% (ie. the x-component of these vectors should be equal to zero).

% Determine the scaling constant for the vector to end in the
% grid's (y,z) plane
c=(d_RT*cos(Azm)+h_LR*cos(Azm)*sin(Aym)+d_GO)/(cos(Aym)*cos(Azm));

% Scale the x-component of the laser's position vector since it is
% aligned with the system's x-axis
p_L=[ c;
      0;
      0;
      1];

% Describe this position vector in the grid's coordinate system
% after it has been scaled. Ensure that the x-component of the
% grid's position vector is zero.
p_G=[p_G T_LGm*p_L];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rotate the subsequent position vectors p_G into the standard
% Cartesian plane with positive Z up and positive Y left
% (All of the position vectors must be flipped across the horizontal
% axis)

% Define the rotation matrix for a 180 degree rotation about the y-axis
RTG=[ cos(pi) 0 sin(pi);
      0 1 0;
      -sin(pi) 0 cos(pi)];

% Initialize the vector to compile the transformed position vectors
p_GRotated=[];

% Loop through the number of rotations
dim=size(p_G);
for k=1:dim(2)

```

```

% Append the transformed position vectors
% (The y and z elements from this vector correspond to the measured
% grid coordinates)
p_GRotated=[p_GRotated RTG*p_G(1:3,k)];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine the actual grid coordinates that track the rifle's
% orientation with the VICON system
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load the rifle's motion data from the VICON system
% This '.mat' file includes the following columns in this order:
% 1. Time stamp from the computer's clock
% 2. X position (in mm)
% 3. Y position (in mm)
% 4. Z position (in mm)
% 5. Angle of rotation about X (in radians)
% 6. Angle of rotation about Y (in radians)
% 7. Angle of rotation about Z (in radians)
% The positions are relative to the platform's reference frame. This
% frame is a right-hand coordinate system with its origin at the near-
% side corner of the platform along the walkway. Its axis directions
% are:
% X -- Along the platform's top horizontal edge heading towards the
% door
% Y -- Along the platform's top horizontal edge heading towards the
% netted wall
% Z -- Up from the platform's top nearest corner along the walkway
load('dataViconBoxNetwork.mat')

% Define the Euler angles in order to convert them into quaternions
% Az: rotation about the z-axis (yaw)(alpha)
% Ay: rotation about the y-axis (pitch)(beta)
% Ax: rotation about the x-axis (roll)(gamma)
% Angles must be in radians ('a' stands for actual)
Aza=Exper_Data2Save(:,7);
Aya=Exper_Data2Save(:,6);
Axa=Exper_Data2Save(:,5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine the position vectors in the grid's coordinate system that
% describe the rifle's orientations. These vectors should end in the
% grid's (y,z) plane (ie. x-component should be zero).

% Loop through the number of rotation angles
% (number of angles should be the same for rotations about Z, Y & X)
p_Ga=[];
laserActual=[];

```

```

Azv=[];
Ayv=[];
Axv=[];
for k=1:length(Aza)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Redefine the transformation matrices with the measured angles of
% rotation from the VICON's readings. These rotations are about
% axis and should be pre-multiplied by their respective coordinate
% fixed systems
% (ie. rotations about the z-axis should be pre-multiplied with the
% tripod's system and rotations about the y-axis should be pre-
% multiplied with the rifle's system).

% Define the special homogeneous transformations for the rotations
% about a fixed axis
Tz=[ cos(Aza(k)) -sin(Aza(k)) 0 0;
     sin(Aza(k))  cos(Aza(k)) 0 0;
     0            0            1 0;
     0            0            0 1];
Ty=[ cos(Aya(k)) 0 sin(Aya(k)) 0;
     0            1 0            0;
    -sin(Aya(k)) 0 cos(Aya(k)) 0;
     0            0 0            1];
Tx=[ 1            0            0 0;
     0 cos(Axa(k)) -sin(Axa(k)) 0;
     0 sin(Axa(k))  cos(Axa(k)) 0;
     0            0            0 1];

% Redefine the description of {L} relative to {G} by pre-multiply
% these rotations by their respective coordinate systems. This
% allows us to determine the resulting position vectors from these
% rotations ('a' stands for actual)
T_LGa=inv(T_G0)*T_TO*Tz*T_RT*Ty*Tx*T_LR;

% Retrieve the Euler angles from R_LGa in order to plot and
% compare to the sensor angles
R_LGa=T_LGa(1:3,1:3)*[cos(pi) sin(pi) 0;-sin(pi) cos(pi) 0; 0 0 1];
[Azr Ayr Axr]=dcm2angle(R_LGa);
Azv=[Azv Azr];
Ayv=[Ayv Ayr];
Axv=[Axv Axr];

% Collect the laser positions in the grid's coordinate system in
% order to calculate the angular error between the sensor's
% readings and the VICON system
laserActual=[laserActual T_LGa(1:3,4)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute the resulting position vectors from these rotations in
% the grid's coordinate system. These vectors should end in the
% grid's (y,z) plane

```

```

% (ie. the x-component of these vectors should be equal to zero).

% Determine the scaling constant for the vector to end in the
% grid's (y,z) plane
c=(d_RT*cos(Aza(k))+h_LR*cos(Aza(k))*sin(Aya(k))+d_G0)/(cos(Aya(k))*cos(Aza(k)
));

% Scale the x-component of the laser's position vector since it is
% aligned with the system's x-axis
p_L=[ c;
      0;
      0;
      1];

% Describe this position vector in the grid's coordinate system
% after it has been scaled. Ensure that the x-component of the
% grid's position vector is zero.
p_Ga=[p_Ga T_LGa*p_L];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rotate the subsequent position vectors p_Ga into the standard
% Cartesian plane with positive Z up and positive Y left
% (All of the position vectors must be flipped across the horizontal
% axis)

% Define the rotation matrix for a 180 degree rotation about the y-axis
RTG=[ cos(pi) sin(pi) 0;
      -sin(pi) cos(pi) 0;
      0 0 1];

% Initialize the vector to compile the transformed position vectors
p_GaRotated=[];

% Loop through the number of rotations
dim=size(p_Ga);
for k=1:dim(2)

    % Append the transformed position vectors
    % (The y and z elements from this vector correspond to the measured
    % grid coordinates)
    p_GaRotated=[p_GaRotated RTG*p_Ga(1:3,k)];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the actual and measured grid coordinates that track the rifle's
% orientation in order to compare the sensor's readings to the ground
% truth data from the VICON system

```

```

% Actual --> VICON Data
% Measured --> Sensor Data
pDesired=[ 0 -30 -30 30 30 0 0;
           0 0 20 20 -10 -10 0];
pActual=[p_GaRotated(2,:);
          p_GaRotated(3,:)];
pMeasured=[p_GRotated(2,:);
            p_GRotated(3,:)];

figure (1)
plot(pDesired(1,:),pDesired(2:3,:), 'r--', pActual(1,:),pActual(2:3,:), 'ms', ...
     pMeasured(1,:),pMeasured(2:3,:), 'bo')
% title('Grids (y,z) Plane')
xlabel('y (cm)')
ylabel('z (cm)')
axis([-50 50 -50 50])
grid on
legend('Desired','VICON','Sensor')
% hold on % for tracking the sensor readings in comparison to the VICON system

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Show how closely the sensor readings track the VICON data in the
% % plots
% figure (1)
% for k=1:10:length(pA(1,:))
%     plot(pA(2,k),pA(3,k),'cd',pM(2,k),pM(3,k),'yv')
%     pause(1)
% end
% hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute and display the distance and angular error between the actual
% and measured grid coordinates
[Error,pA,pM,kAStart,kAStop,kMStart,kMStop]=sensorerror(pActual,pMeasured,lase
rActual,laserMeasured);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the truncated grid coordinates from the VICON system and the
% sensor in order to compare when both sets of data are being truncated

figure (2)
plot(pA(2,:),pA(3,:), 'ms', pM(2,:),pM(3,:), 'bo')
% title('Grids (y,z) Plane')
xlabel('y (cm)')
ylabel('z (cm)')
axis([-50 50 -50 50])
grid on
legend('VICON','Sensor')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the distance errors
figure (3)

```

```

plot(Error(1,:), 'r', 'LineWidth', 1.5)
% title('Distance Error between the Actual and Measured Coordinates')
xlabel('Digital Freq \omega')
ylabel('Distance Error (cm)')
xlim([1 length(Error(1,:))])
grid on

% Plot the angular errors
figure (4)
plot(Error(2,:), 'r', 'LineWidth', 1.5)
% title('Angular Error between the Actual and Measured Coordinates')
xlabel('Digital Freq \omega')
ylabel('Angular Error (degree)')
xlim([1 length(Error(1,:))])
grid on

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the actual and measured Euler angles in order to compare the
% VICON system's and sensor's data in the {L} coordinate system
% Az: rotation about the z-axis (yaw)(alpha)
% Ay: rotation about the y-axis (pitch)(beta)
% Ax: rotation about the x-axis (roll)(gamma)

% Resize the actual and measured angles since the VICON system and
% sensor sample at different rates in order to compare the angles as
% closely as possible
[Azv, Azs, wzv, wzs] = samplerate(Azv, Azs, kAstart, kAstop, kMstart, kMstop);
[Ayv, Ays, wyv, wys] = samplerate(Ayv, Ays, kAstart, kAstop, kMstart, kMstop);
[Axv, Axs, wxv, wxs] = samplerate(Axv, Axs, kAstart, kAstop, kMstart, kMstop);

% Plot the Euler angles
figure (5)
subplot(3,1,1)
plot(wzv, Azv*180/pi, 'b', 'LineWidth', 1.5)
hold on
plot(wzs, Azs*180/pi, 'r', 'LineWidth', 1.5)
ylabel('Yaw \alpha')
xlim([0 2*pi])
% title('Actual and Measured Euler Angles in Degrees')
legend('VICON', 'Sensor')
hold off
subplot(3,1,2)
plot(wyv, Ayv*180/pi, 'b', 'LineWidth', 1.5)
hold on
plot(wys, Ays*180/pi, 'r', 'LineWidth', 1.5)
ylabel('Pitch \beta')
xlim([0 2*pi])
hold off
subplot(3,1,3)
plot(wxv, Axv*180/pi, 'b', 'LineWidth', 1.5)
hold on
plot(wxs, Axs*180/pi, 'r', 'LineWidth', 1.5)
ylabel('Roll \gamma')

```

```
xlim([0 2*pi])
xlabel('Digital Freq \omega [0,2\pi]')
hold off
```

C. PYTHON SCRIPTS FOR TESTING THE TRACKING ALGORITHM

The Python programming language was used to implement the geometry-of-fire tracking algorithm. This language was selected since it was compatible with the network's hardware components. A function was written to publish node positions, and the main program called this function and implemented the tracking algorithm. It is important to note that the main program was developed from open-source templates for Python code. This template was found by ENS Jonathan Driesslein, USN, who developed the tactical network for this specific application. All modifications to this template are indicated with comments.

1. Function for Publishing Node Positions

```
#!/usr/bin/env python
#license removed for brevity

import rospy
from geometry_msgs.msg import Quaternion

def talker():
    pub = rospy.Publisher('/NodePositions', Quaternion, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        data.x = 450;
        data.y = 150;
        data.z = -34;
        data.w = 10;
        rospy.loginfo(data)
        pub.publish(data)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

2. Script File for Implementing Tracking Algorithm

```
#!/usr/bin/env python

import zmq
from threespace_api import TSDLSensor as ts
import threespace_api
from threespace_api import *
import serial
import time
import math
import string
import colorsys
import rospy
import roslib
import threading
from std_msgs.msg import String
from geometry_msgs.msg import Quaternion
from socket import *
from numpy import * # Added for Our Code
from numpy import linalg # Added for Our Code
#import json_utils
#import nmea

#####
##### Start Our Code #####
#####

#####
# Define global variables that must be used in the main
# script and functions
global posFriendly
posFriendly=matrix([[ 0, 0, 0, 0],
                    [ 0, 0, 0, 0],
                    [ 0, 0, 0, 0]]) #Negate the sign for opposite direction

##### ROS #####
# Define functions that will be called by the main script

def PosePub():
    pub = rospy.Publisher('RifleData', String, queue_size = 20) #'RifleData'
    will be the name of the node
    rospy.init_node('PosePub', anonymous=True) #I think that 'PosePub' will
    be what we call it in the program
    rate = rospy.Rate(5) #5hz
    if not rospy.is_shutdown():
        PubOut = "Roll [%.2f] Pitch [%.2f] Yaw [%.2f]" % (roll, pitch,
yaw)
        rospy.loginfo(PubOut)
        pub.publish(PubOut)
        rate.sleep()
```



```

#         rospy.spin()

def callback(data):
    global posFriendly
    #rospy.loginfo(rospy.get_caller_id() + "I heard %s," data)
    posFriendly[:,data.w]=matrix([[data.x],[data.y],[data.z]])
    #print(posFriendly)
    return posFriendly
    #"I heard X: [%.2f]\n Y: [%.2f]\n Z: [%.2f]\n," (data.x, data.y,
data.z))

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # name are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    #rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("NodePositions", Quaternion, callback)

    # spin() simply keeps python from exiting until this node is stopped
    #rospy.spin()

#
#Defining the thread that will be listening to ROS Topic
#

listThread = threading.Thread(target=listener)
listThread.daemon = True

#####
# Function: Determine if any of the friendly nodes are
# within the safety threshold surrounding the rifle's
# cone of fire

def detectfriendly(riflePose,posRifle,posFriendly):

# INPUT:
#   riflePose -- Rifle's laser orientation in the NED
#               coordinate system. Expect a 3x1 matrix
#               for the (x,y,z) components from the
#               position vector (in centimeters)
#   posRifle -- Rifleman's position in the NED coordinate
#               system. Expect a 3x1 matrix for the
#               (x,y,z) components (in centimeters)
#   posFriendly -- Friendlies' positions in the NED
#                  coordinate system. Expect a 3xn
#                  matrix for the (x,y,z) components
#                  where n is the number of friendly
#                  positions that are passed for that
#                  time step (in centimeters)
# OUTPUT: Print 'CEASE FIRE: Geometries of fire must be

```

```

# de-conflicted' if a friendly position is within the
# safety threshold

# Loop through the number of friendly positions
i=0;
while i<posFriendly.shape[1]:

    # Compute the position vector that describes the
    # friendly's position with respect to the rifleman
    pFriendly=posFriendly[:,i]-posRifle

    # Skip friendly positions that are within a meter of
    # the rifleman's position
    normpFriendly=linalg.norm(pFriendly)
    if normpFriendly>100:
        print(normpFriendly)
        # Compute the angle between the friendly's position
        # vector and the rifle's pose (in radians)
        dotProd=dot(transpose(pFriendly),riflePose)
        crossProd=cross(pFriendly,riflePose,axis=0)
        normCross=linalg.norm(crossProd)
        angle=arctan2(normCross,dotProd)

        # Convert the angle from radians to degrees
        angle=angle*180/math.pi

        # Compare the angle to the safety threshold
        # Note: the safety threshold is approximately 8.5
        # degrees around the muzzle of the rifle which
        # corresponds to 300 mils
        if abs(angle)<=16.9:

            # Print the geometries of fire warning
            # and break out of the loop

            print("*****")
            print("CEASE FIRE: Geometries of fire must be
de-conflicted!")
            print("Angle: [%.2f]" % angle)

            print("*****")
            break

    # Increment the index
    i+=1

#####
##### End Our Code #####
#####

#BROADCASTPORT=50001 #the port for NMEA PASHR data

```

```

counter = 0

#port = "COM3" #for windows
port = "/dev/ttyACM0" #for Raspberry Pi

try:
    print("Opening with ThreesSpace API")
    device = threespace_api.TSDLSensor(com_port=port) #this works. Yippee
    print(device)
    print("Yippee. Found device # {0}."format(device))
except:
    ## If a connection to the COM port fails, None is returned.
    print("Boo. No device on {0}."format(port))
    #print(device)
    #exit()
else:
    if device is not None:
        #configure the UDP socket for broadcast
        #cs = socket(AF_INET, SOCK_DGRAM)
        #cs.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
        #cs.setsockopt(SOL_SOCKET, SO_BROADCAST, 1)

        #configure the ZeroMQ so we can send data to the website
        #context = zmq.Context()
        #pub = context.socket(zmq.PUB)
        #pub.setsockopt(zmq.HWM, 1)
        #pub.connect("epgm://127.0.0.1:5000")

#####
##### Start Our Code #####
#####

#####
# List of the coordinate systems
# {0}: NED
# {T}: Tripod (rotations about z)
# {R}: Rifle (rotations about y and x)
# {L}: Laser
# {S}: Sensor
# {A}: Actual

#####
# Define the constant variables for the system model

# Define the rifleman's position (in centimeters) in
# the {0} coordinate system that is being passed to
# the network. Note: the {T} coordinate system
# represents the rifleman in this script.
O_T0=matrix([[0], [0], [0]]) # Rifleman's position {T} relative to {0}

# Define the friendlies' positions (in centimeters) in
# the {0} coordinate system that are being passed from
# the network
#posFriendly=matrix([[ 0, 0, 100],

```

```

#           [ 0, 100, 0],
#           [-100, 0, 0]]) #Negate the sign for opposite
direction

# Define the offsets (in centimeters) between the coordinate systems
h_TO=92 # Hieght of the {T} origin from the {O} origin
h_RT=7 # Hieght of the {R} origin from the {T} origin
h_LR=17 # Hieght of the {L} origin from the {R} origin
h_SL=4 # Hieght of the {S} origin from the {L} origin
d_RT=2.5 # Distance of the {R} origin from the {T} origin
d_SL=42 # Distance of the {S} origin from the {L} origin

# Specify the Euler angles (in radians) for the reference
# position. Note: the reference position has the rifle
# level and oriented in the opposite direction of the
# drawn {O} x-direction
Az=180*math.pi/180 # rotation about z-axis (yaw)(alpha)
Ay=0*math.pi/180 # rotation about y-axis (pitch)(beta)
Ax=0*math.pi/180 # rotation about x-axis (roll)(gamma)

# Define the transformation matrices between the coordinate
# systems that are fixed for the reference position
T_TO=matrix([[-cos(Az), sin(Az), 0, O_TO.item(0)],
              [-sin(Az), -cos(Az), 0, O_TO.item(1)],
              [0, 0, 1, O_TO.item(2)-h_TO],
              [0, 0, 0, 1]]) #
Description of {T} relative to {O}

T_RT=matrix([[cos(Ay), 0, sin(Ay), -d_RT],
              [0, 1, 0, 0],
              [-sin(Ay), 0, cos(Ay), -h_RT],
              [0, 0, 0, 1]]) #
Description of {R} relative to {T}

T_LR=matrix([[1, 0, 0, 0],
              [0, 1, 0, 0],
              [0, 0, 1, -h_LR],
              [0, 0, 0, 1]]) #
Description of {L} relative to {R}

T_LR=matrix([[1, 0, 0, -d_SL],
              [0, 1, 0, 0],
              [0, 0, 1, h_SL],
              [0, 0, 0, 1]]) #
Description of {L} relative to {R}

# Define the compound transformations between the coordinate
# systems that are not directly fixed to each other
T_LO=T_TO*T_RT*T_LR # Description of {L} relative to {O}

# Define the laser's position vector in {L} for the
# reference position
pr_L=matrix([[50000],[0],[0],[1]]) # 50000cm = 500m (rifle's maximum
effective range)

```

```

# Describe this reference position vector in {0}
pr_0=T_L0*pr_L

#####
##### End Our Code #####
#####

#Fire up listening thread
listThread.start()
while (True):

    #turn the LED off
    #color = [0, 0, 0]
    #threespace_api.TSDLSensor.setLEDColor(device,color)

    ##quat = ts.getTaredOrientationAsQuaternion(device)
    quat=
    threespace_api.TSDLSensor.getUntaredOrientationAsQuaternion(device)
    if quat is not None:
        x = quat[0]
        y = quat[1]
        z = quat[2]
        w = quat[3]
        #http://answers.unity3d.com/questions/416169/finding-
pitchrollyaw-from-quaternions.html

#####
##### Start Our Code #####
#####

#####
# Convert the quaternion from the sensor's reading to
# Euler angles
# Note: Referencing MATLAB code given to Caleb Khan from
# James Calusdian. Original author is Xiaoping Yun.

# If the quaternion scalar component w is negative,
# then negate the sign of the quaternion vector component
if w<0:
    x=-x
    y=-y
    z=-z
    w=-w
print(x,y,z,w)
B=matrix([[w**2+x**2-y**2-z**2, 2*(x*y+z*w), 2*(x*z-w*y)],
          [2*(x*y-w*z), w**2-x**2+y**2-z**2, 2*(y*z+w*x)],
          [2*(x*z+w*y),2*(y*z-w*x),w**2-x**2-y**2+z**2]]);

roll=math.atan2(B[1,2],B[2,2])*180/math.pi
pitch=-math.asin(B[0,2])*180/math.pi
yaw=math.atan2(B[0,1],B[0,0])*180/math.pi

```

```
#####
# Assumptions: Assume that the rifleman is positioned with
# his or her rifle (and sensor) initially aligned with the
# actual NED's positive x-axis. An offset rotation matrix
# must be defined in the future to account for the sensor
# not being aligned as described.

#####
# Adjust the rifle's laser orientation based on the
# rotations that correspond to the sensor's readings from
# above

# Convert the Euler angles into radians
# Note: 'm' stands for measured angles
Azm=yaw*math.pi/180 # rotation about z-axis (yaw)(alpha)
Aym=pitch*math.pi/180 # rotation about y-axis (pitch)(beta)
Axm=roll*math.pi/180 # rotation about x-axis (roll)(gamma)

# Define the special homogeneous transformations for
# the rotations about a fixed axis
Tz=matrix([[cos(Azm), -sin(Azm), 0, 0],
           [sin(Azm), cos(Azm), 0, 0],
           [0, 0, 1, 0],
           [0, 0, 0, 1]]) # Rotation about the z-
axis

Ty=matrix([[cos(Aym), 0, sin(Aym), 0],
           [0, 1, 0, 0],
           [-sin(Aym), 0, cos(Aym), 0],
           [0, 0, 0, 1]]) # Rotation about the y-
axis

Tx=matrix([[1, 0, 0, 0],
           [0, cos(Axm), -sin(Axm), 0],
           [0, sin(Axm), cos(Axm), 0],
           [0, 0, 0, 1]]) # Rotation about the x-
axis

# Redefine the description of {L} relative to {0} by
# pre-multiplying these rotations by their respective
# coordinate systems
# Note: 'm' stands for measured transformations
T_LOm=T_TO*Tz*T_RT*Ty*Tx*T_LR

# Compute the resulting position vectors that describes
# the rifle's laser orientation after these rotations
# in {0}
pm_0=T_LOm*pr_L

# Determine if any of the friendly nodes are
# within the safety threshold surrounding the rifle's
# cone of fire
# if __name__ == '__main__':
#     try:
```

```

        # listener()
    # except rospy.ROSInterruptException:
        # pass
    detectfriendly(pm_0[[0,1,2],:],O_T0[[0,1,2],:],posFriendly)

    # Display the friendly's position and rifle's position
    # vector for comparison in order to verify the 'cease fire'
    # print statement in the detectfriendly() function
    print(posFriendly) # Friendly's (x,y,z) position
    pm_ONorm=math.sqrt(pm_0[0]**2+pm_0[1]**2+pm_0[2]**2)
    pm_OScaled=pm_0[[0,1,2],:]/pm_ONorm
    pm_OScaled=100*pm_OScaled
    print(pm_OScaled) # Rifle's scaled position vector

#####
##### End Our Code #####
#####

##### ROS #####

        #print("Roll  [%.2f] Pitch  [%.2f] Yaw  [%.2f]" % (roll, pitch,
yaw))
            #print("Pitch [%.2f]" % pitch)
            #print("Yaw [%.2f]" % yaw)
    if __name__ == '__main__':
        try:
            PosePub()
        except rospy.ROSInterruptException:
            pass

    #now flash the LED so we know it is communicating, but how??? - ask
YEI support
    color = [0, 0, 255]
    time.sleep(0.04)
    threespace_api.TSDLSensor.setLEDColor(device,color)
    time.sleep(0.04)
    counter += 1
    #exit(1)

#####

```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- [1] Rocketbox Libraries. (2012, Sept. 5). High-end military 3D characters and animations by Rocketbox. [Online]. Available: <http://rocketbox-libraries.com/us/index.php/hdsoldierspress>.
- [2] *Marine Corps Planning Process*, Marine Corps Warfighting Publication 5-1, USMC, 2000, pp. 2–5.
- [3] J. Calusdian, “A personal navigation system based on inertial and magnetic field measurements,” Ph.D. dissertation, Dept. Elect. & Comp. Eng., Naval Postgraduate School, Monterey, CA, 2010.
- [4] J. Rantakokko, J. Rydell, P. Stromback, P. Handel, J. Callmer, D. Tornqvist, F. Gustafsson, M. Jobs, and M. Gruden, “Accurate and reliable soldier and first responder indoor positioning: Multisensor systems and cooperative localization,” *IEEE Wireless Communications*, vol. 18, no. 2, pp. 10–17, April 2011.
- [5] J. C. Driesslein, “Scalable Mobile Ad Hoc Network (MANET) to Enhance Situational Awareness in Distributed Small Unit Operations,” M.S. thesis, Dept. Elect. & Comp. Eng., Naval Postgraduate School, Monterey, CA, 2015.
- [6] *Marine Rifle Squad*, Marine Corps Warfighting Publication 3-11.2, USMC, 2002, 9.1.
- [7] *Crew-Served Machine Guns, 5.56-mm and 7.62-mm*, FM 3-22.68, U.S. Dept. of the Army, Washington, DC, 2003, 5.3–5.4.
- [8] USGI Colt carbine parts variation guide Edition IV. (n.d.). AR15.com. [Online]. Available: <https://www.ar15.com/mobile/topic.html?b=3&f=123&t=296919>. Accessed Aug. 17, 2015.
- [9] R. Lammie. (n.d.). Give me a sign: The stories behind 5 hand gestures. [Online]. Available: <http://mentalfloss.com/article/24475/give-me-sign-stories-behind-5-hand-gestures>. Accessed Aug. 17, 2015.
- [10] J. J. Craig, “Introduction” in *Introduction to Robotics: Mechanics and Control*, 2nd ed. Reading, MA: Addison-Wesley, 1989, ch. 1, sec. 2–3, pp. 4–16.
- [11] J. J. Craig, “Spatial Descriptions and Transformations” in *Introduction to Robotics: Mechanics and Control*, 2d ed. Reading, MA: Addison-Wesley, 1989, ch. 2, pp. 19–67.
- [12] M4 carbine. (2015, Aug. 15). *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/M4_carbine. Accessed Aug. 17, 2015.

- [13] AN/PEQ-15. (n.d.). U.S. Night Vision. [Online]. Available: <http://www.usnightvision.com/anpeq-15atpial.aspx>. Accessed Aug. 17, 2015.
- [14] Video camera tripods. (n.d.). MediaCollege.com. [Online]. Available: <http://www.mediacollege.com/video/camera/tripod/>. Accessed Aug. 17, 2015.
- [15] 3-Space data-logging. (n.d.). YEI Technology. [Online]. Available: <https://www.yeitechnology.com/productdisplay/3-space-data-logging>. Accessed Mar. 24, 2015.
- [16] J. B. Kuipers, “Algorithm Summary,” in *Quaternions and Rotation Sequences*, Princeton, NJ: Princeton Univ. Press, 1999, ch. 7, sec. 7, pp. 167–168.
- [17] YEI 3-space sensor. (n.d.). YEI Technology. [Online]. Available: http://www.yeitechnology.com/sites/default/files/TSS_Family_Tech_Brief_v1.0.4b.pdf. Accessed Mar. 24, 2015.
- [18] VICON. (n.d.). VICON Motion Systems. [Online]. Available: <http://www.vicon.com/>. Accessed Jun. 10, 2015.
- [19] A. A. Grompone, “Vision-based 3D motion estimation for on-orbit proximity satellite tracking and navigation,” M.S. thesis, Dept. Elect. & Comp. Eng., Naval Postgraduate School, Monterey, CA, 2015.
- [20] “Logic functions of two variables,” class notes for Digital Logic Circuits, Dept. Elect. & Comp. Eng., Naval Postgraduate School, Monterey, CA, fall 2015.
- [21] 3-Space sensor user’s manual. (2014, Nov. 14). YEI Technology. [Online]. Available: http://www.yeitechnology.com/sites/default/files/YEI_TSS_Users_Manual_3.0_r1_4Nov2014.pdf. Accessed Jul. 18, 2015.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California