DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DESIGNING A SENSOR FUSION ALGORITHM FOR POSITION AND ORIENTATION ESTIMATION USING AN IMU AND A POSITION SENSOR, AND ITS IMPLEMENTATION

by

Ekrem YAVUZ

March, 2024

İZMİR

DESIGNING A SENSOR FUSION ALGORITHM FOR POSITION AND ORIENTATION ESTIMATION USING AN IMU AND A POSITION SENSOR, AND ITS IMPLEMENTATION

A Thesis Submitted to the
Graduate School of Natural And Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Department of Mechatronics Engineering

by

Ekrem YAVUZ

March, 2024
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**DESIGNING A SENSOR FUSION ALGORITHM FOR POSITION AND ORIENTATION ESTIMATION USING AN IMU AND A POSITION SENSOR, AND ITS IMPLEMENTATION**" completed by **EKREM YAVUZ** under supervision of **ASSOC. PROF. DR. YAVUZ ŞENOL** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

.....

Assoc. Prof. Dr. Yavuz ŞENOL

Supervisor

.....	.....
Prof. Dr. Uğur ÇAM	Assoc. Prof. Dr. Aytaç GÖREN
Thesis Committee Member	Thesis Committee Member
.....	.....
Prof. Dr. H. Metin ERTUNÇ	Assoc. Prof. Dr. Barış BIDIKLI
Examining Committee Member	Examining Committee Member

Prof. Dr. Okan FISTIKOĞLU
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

First and foremost, I owe my deepest gratitude to my supervisor, Assoc. Prof. Dr. Yavuz ŞENOL, for the valuable information he shared throughout my doctoral education, his guidance in navigating through challenging situations, his efforts to keep my motivation high, and his consistently patient and understanding attitude. Getting to know him and working with him is a great honor and privilege.

Secondly, I express sincere gratitude to my thesis committee members, Prof. Dr. Uğur ÇAM and Assoc. Prof. Dr. Aytaç GÖREN, for their valuable contributions and the time they devoted to my thesis progress meetings.

Furthermore, I would like to thank Mr. Hüseyin AYDIN and Mr. Merih ÖZÇELİK, who contributed to this study in every aspect and worked tirelessly alongside me for a very long time.

Moreover, I am very grateful to my dear mother Nursel YAVUZ and dear father Rasim YAVUZ, who have put in great effort for me to reach these days.

Finally, I extend my heartfelt thanks to my dear spouse, M.D. Berna AVCI YAVUZ, for her unwavering support throughout my doctoral education and for encouraging me in successfully completing my thesis work.

Ekrem YAVUZ

DESIGNING A SENSOR FUSION ALGORITHM FOR POSITION AND ORIENTATION ESTIMATION USING AN IMU AND A POSITION SENSOR, AND ITS IMPLEMENTATION

ABSTRACT

The starting point of this study is to address and seek a solution to a challenge in the textile industry. Potassium permanganate is primarily used for denim lightening in the textile industry. The spray application of microparticles of PP to denim causes long-term health risks for workers, as inhaling them may lead to lung diseases. The idea is that robots can be employed instead of individuals in this spraying process. This led us to the concept of recording the position and orientation, i.e. the pose, of the spray gun during the spraying process, which is performed by a skilled operator. Subsequently, these movements can be accurately replicated by robots without compromising human health.

To determine the pose of an object in space with high precision, it is necessary to establish sensor systems and utilize algorithms that can interpret the data obtained from sensors. Therefore, the primary focus of this thesis is the development of a sensor fusion algorithm and sensory system designed for six-dimensional pose estimation, followed by controlling a robot arm using recorded pose data for PP spray applications.

Numerous methods and sensors are available for determining the pose of and object. The proposed approach involves the integration of data from an Inertial Measurement Unit and string-encoder based position sensors to enhance the accuracy of pose estimations.

The designed algorithm benefits from features of filter types, such as Kalman filters and Complementary filters, aiming to address their shortcomings and improve the precision of Euler angle calculations. Additionally, in the study, the results obtained when various sensor fusion filter types were tested with the same data have also been compared.

As a result, this thesis has proven and practically demonstrated in the created test setup that the pose of an object in space can be precisely determined by leveraging the power of sensor fusion algorithms through the combination of IMU and string encoders.

Keywords: Sensor fusion, pose measuring, movement record, offline robot programming, robot teleoperation, sensory system, string-encoder, inertial measurement unit, tool tracking system.

IMU VE KONUM SENSÖRÜ KULLANARAK KONUM VE ORYANTASYON TESPİTİ YAPAN SENSÖR FÜZYON ALGORİTMASI TASARLAMAK VE UYGULAMASI

ÖZ

Bu çalışmanın başlangıç noktası, tekstil endüstrisinde karşılaşılan bir probleme çözüm aramaktır. Potasyum permanganat, bir inorganik kimyasal bileşen ve oksidasyon ajanı olarak, özellikle tekstil endüstrisinde denim renk açma işlemi için kullanılmaktadır. PP mikropartiküllerinin denim üzerine püskürtülmesi, işçiler için uzun vadede sağlık riskleri oluşturur, çünkü bu partikülleri solumak akciğer hastalıklarına yol açabilir. Bu duruma bir çare olarak, püskürtme işleminde insanlar yerine robotların kullanılabileceği fikri ortaya çıkmıştır. Bu bizi, bir uzman operatör tarafından püskürtme işlemi sırasında püskürtme tabancasının pozisyonunu ve yönelimini, yani pozunu, kaydetme konseptine yönlendirdi. Daha sonra, bu hareketler, insan sağlığını tehlikeye atmadan robotlar tarafından kusursuz bir şekilde taklit edilebilecek ve püskürtme işlemi yapılabilecektir.

Bir nesnenin uzaydaki pozunu yüksek bir hassasiyetle belirlemek için sensör sistemleri kurmak ve sensörlerden elde edilen verileri yorumlayabilen algoritmaları kullanmak gereklidir. Bu nedenle, bu tezin temel odak noktası, altı boyutlu poz tahmini için tasarlanmış bir sensör füzyon algoritması geliştirilmesi ve sensör sistemi kurulmasıdır. Ardından, kaydedilen poz verileri kullanılarak bir robot kolu kontrol etme sürecini içermektedir.

Bir nesnenin pozunu belirleme için birçok yöntem ve sensör tipi bulunmaktadır. Önerilen yaklaşım, poz tahminlerinin doğruluğunu artırmak amacıyla bir İnersiyal Ölçme Ünitesi (IMU) ve tel kodlayıcı tabanlı pozisyon sensörlerinden elde edilen verilerin birlikte kullanılmasını içerir.

Geliştirilen algoritma, Kalman filtreleri ve Complementary filtreler gibi filtre türlerinin özelliklerinden faydalanarak, eksikliklerini ele almaya ve Euler açı

hesaplamalarının doğruluğunu artırmaya çalışır. Ayrıca çalışmada, bir çok farklı sensör füzyon filtre türünün aynı veri ile test edildiğinde ortaya çıkardıkları sonuçlar da karşılaştırılmıştır.

Sonuç olarak, bu tezde, IMU ve tel kodlayıcıların birlikte kullanılması ve sensör füzyon algoritmaları sayesinde uzaydaki bir nesnenin pozunun kesin bir şekilde belirlenebileceği teorik olarak kanıtlanmış ve oluşturulan test düzeneğinde uygulamalı olarak gösterilmiştir.

Anahtar kelimeler: Sensor füzyonu, poz ölçümü, hareket kayıt, çevrimdışı robot programlama, robot teleoperasyonu, sensör sistemleri, tel kodlayıcı pozisyon sensörü, inersiyal ölçme ünitesi, aygıt takip sistemi.

LIST OF SYMBOLS

$\hat{x}_k$	: Estimated state
A	: State transition matrix
x_k	: The state of the system at time k
P_k	: Error covariance matrix
A^T	: Transpoze of state transition matrix
Q	: Process noise covariance
K_k	: Kalman gain
H^T	: Transpoze of the observation matrix
H	: Observation matrix
R	: Measurement noise covariance
z_k	: Measurement vector
f	: Nonlinear state transition function
u_k	: Control input
h	: Measurement function
z_k	: Time vector
$\theta_{complementary}$	: Final orientation estimation
θ_{Acc}	: Angle computed from accelerometer data
θ_{Gyro}	: Angle computed from gyroscope data
α	: Tuning parameter
u_k	: Control input
q	: Quaternion
ω	: Angular rate vector from the gyroscope

β	: Filter parameter
B	: Control input matrix
phi_dot	: Rate of change of the roll angle
theta_dot	: Rate of change of the pitch angle
phi_hat_acc	: Observed roll
theta_hat_acc	: Observed pitch
y_tilde	: Innovation or measurement residual
S	: innovation covariance

ABBREVIATIONS

PP	:Potassium Permanganate
IMU	:Inertial Measurement Unit
KMnO4	:Potassium permanganate
HMI	:Human-Machine Interface
PbD	:Programming by Demonstration
MEMS	:Micro Electro Mechanical Systems
GPS	:Global Positioning System
MARG	:Magnetic, Angular Rate, and Gravity
PDR	:Personal Dead Reckoning
DCM	:Direct Cosine Matrix
DOF	:Degrees of Freedom
EKF	:Extended Kalman Filter
TCP	:Tool Center Point
CAD	:Computer Aided Design
3D	:three dimension
PCB	:Printed Circuit Board
XKF3	:Xsens Kalman Filter
SDK	:Software Development Kit
LLCP	:Low Level Communication Protocol
GPIO	:General Purpose Input/Output
GUI	:Graphical User Interface
C-API	:C Application Programming Interface

RTMachine	:Runtime Machine
TCP/IP	:Transmission Control Protocol / Internet Protocol
RTDE	:Real-Time Data Exchange
PLC	:Programmable Logical Controller
LD	:Ladder Diagram
PC	:Personel Computer
MDK	:Microcontroller Development Kit
SPI	:Serial Peripheral Interface
I/O	:Input/Output
USB	:Universal Serial Bus
IDE	:Integrated Development Environment
SMC	:Sequential Monte Carlo
RF	:Radio Frequency
EO/IR	:Electro-Optical/Infrared
TTF	:Teaching Tool Front
TTB	:Teaching Tool Back
HEX	:Hexadecimal
MSB	:Most Significant Bit
XDA	: Xsens Device API
RMSE	:Root Mean Square Error
AHRS	:Attitude and Heading Reference System

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

1.1 Introduction

Potassium permanganate (KMnO4) is a chemical compound consisting of potassium, manganese, and oxygen, manifests as dark purple crystals, recognized for its potent oxidizing capabilities and has various applications in different fields. In the realm of water treatment, it assumes a pivotal role in the purification process, effectively eliminating impurities and organic matter. Furthermore, its efficacy as an antiseptic renders it instrumental in medical contexts, particularly for disinfecting wounds and addressing various skin ailments. Within the ambit of chemical synthesis, potassium permanganate finds utility as a reagent in laboratory settings, facilitating the preparation of organic compounds. Environmental remediation endeavors leverage its oxidation potential for the treatment of contaminated soil and groundwater. In fisheries and aquaculture, it serves to mitigate parasites and infections in aquatic organisms. In analytical chemistry, it is employed as a reagent for titrations and other chemical analyses.

The area of interest for us is, the compound plays a crucial role in the textile industry as a potent bleaching agent. In denim production, particularly, it achieves the distressed or worn-out appearance synonymous with modern fashion trends. The oxidative properties of potassium permanganate facilitate the removal of color from fabrics, creating intentional variations and patterns on denim surfaces. The controlled application of this chemical during the bleaching process allows textile manufacturers to customize the level of distress, contributing to the unique and fashionable finishes observed in denim garments. The utilization of potassium permanganate as a bleaching agent underscores its significance in shaping the aesthetic attributes of textiles within the dynamic landscape of the fashion industry.

However, alongside all these beneficial features, potassium permanganate poses a potential threat to human health due to its strong oxidizing properties. The spraying

process of potassium permanganate may result in the inhalation of chemical microparticles by workers, thereby posing long-term risks of respiratory disorders. Additionally, direct contact with the compound can induce skin and mucous membrane irritation. Hence, it is imperative to minimize workers' exposure to potassium permanganate. In this case, industrial robots become an attractive option that can replace workers and carry out the same task at a level of proficiency at least equal to theirs.

Industrial robots have become indispensable assets in modern manufacturing, playing a crucial role in enhancing productivity, precision, and efficiency across various industries. These automated machines are designed to execute tasks with accuracy and consistency, reducing human error and increasing overall production quality. The versatility of industrial robots enables their application in diverse fields, including automotive assembly lines, electronics manufacturing, pharmaceutical production, and more. Their ability to handle repetitive and hazardous tasks not only contributes to workplace safety but also allows human workers to focus on more complex and creative aspects of production. As technology continues to advance, industrial robots are evolving to incorporate features such as artificial intelligence and machine learning, further expanding their capabilities and adaptability. The increasing integration of these robots into various industrial processes underscores their pivotal role in shaping the future of manufacturing.

Utilizing a robotic system for the spraying process can prove to be a highly efficacious solution. The integration of robotic technology mitigates potential health risks associated with manual handling, offering a safer and more controlled environment. In addition, the use of robots in the spraying process offers fast production and better product standardization.

Robots require programming before they can be utilized in any desired process. Several approaches exist for programming robots, with the prevalent method involving the use of a teach pendant (Gao et al., 2015). The teach pendant is a human-machine interface (HMI) that is used for real-time monitoring of the robots,

requiring manual definition by the operator of various parameters, i.e. motion modes, speeds, target points, commands and functions. Using a teach pendant for programming a robot is efficient for tasks like welding and screwing, which only require teaching a few points to the robot. However, applications like spraying, surface finishing, and flame cutting, which involve continuous paths, programming by a teach pendant is inadequate. In this thesis, our focus was directed towards the Programming by Demonstration (PbD), a technique designed to empower individuals with the ability to program robots without the necessity of prior programming knowledge. The demonstration operation can occur through two main methods, namely kinesthetic or teleoperation (Akgun et al., 2012). In kinesthetic demonstration, a human teacher physically guides the robot, while teleoperation teaching eliminates the need for physical guidance. This method allows the teacher and the robot to be physically distant, with the teacher not requiring direct visibility of the robot. A teleoperation device, known as the teaching tool, establishes a connection between the human teacher and the robot, enabling the robot's end-of-arm tool to mimic the teacher based on the teleoperation device's movements. The primary objective of this method is to enable the robot to replicate movements performed by a human teacher, no matter what is the operator's familiarity with robot programming or robotics.

The continuous recording of the teaching tool's movements, carried out by a tracking system, is essential to successfully imitating the movements. A nearly error-free movement recording can be achieved through the thoughtful design of both the tracking system and the teaching tool, specifically customized for the task. In this thesis, the pose of the spray gun had to be continuously recorded during the teaching process.

Trajectory tracking systems presented in existing studies can be classified based on the type of sensor used such as acoustic, electromagnetic, microelectromechanical (MEMS), optical, radiofrequency etc. (Božek et al., 2017; Kilin et al., 2017). Each sensor type has its advantages and disadvantages, and the selection depends on the specific application requirements. For example, electromagnetic-based tracking

systems may not be suitable in areas with strong magnetic field sources (Kirsch et al., 2006), while vision-based tracking systems offer high accuracy but can be affected by external factors such as light intensity, occlusion, or reflection (Moeslund & Granum, 2001). Laser tracking systems provide excellent accuracy but involve a lengthy calibration process and are often expensive and sensitive to external influences (Aoyagi et al., 2010).

A suitable method for implementation in this thesis involves string-encoder-based tracking systems, known for providing high accuracy and noise-free outputs with ease of implementation. Nevertheless, these systems exhibit minor structural imperfections, including dimensional variations depending on production tolerances, uncertainties in string length, or slackness in the strings (Thomas et al., 2005). The increase in the number of strings corresponds to an increase in the force exerted on the movable object, thereby causing challenges in achieving high-speed movement of the object (Merlet, 2008; Thomas et al., 2002).

Another prominent tracking device is the MEMS-based IMU, comprising a triaxial accelerometer (linear acceleration), a triaxial gyroscope (angular velocity), and in some instances, a triaxial magnetometer (magnetic flux density) to enhance precision. IMUs being widely preferred due to their cost-effectiveness and applicability in indoor settings, unlike GPS-enabled systems. Despite various advantages, a key limitation of the IMU is its inability to sustain long-term tracking. The accumulation of noisy sensor data during integration leads to increased output errors over time (Won et al., 2009a). Error types of IMU can be classified into two, which are random errors and systematic errors. While systematic errors can be eliminated through calibration methods, random errors, comes from sensor unpredictability, cannot be completely eliminated and their impact on the results are more than systematic errors. (Tomaszewski et al., 2015). Moreover, mechanical vibrations and shocks, temperature variations and magnetic interferences contribute to IMU performance variability. Combining IMUs with different type of sensors is a common strategy to mitigate these challenges and enhance overall system accuracy.

Hybrid systems, integrating an IMU with a position sensor, find applications in diverse areas for enhanced accuracy in tracking the pose of moving objects. Position sensors such as ultrasonic (Zhao & Wang, 2012), radiofrequency-based (Zhang et al., 2020), odometer (Pirník et al., 2015; Georgy et al., 2011), optical (Atac & Foxlin, 2013), or vision-based (Nützi et al., 2011; Alatise & Hancke, 2017) systems can be combined with IMUs. In addition to these sensors, it is widely preferred to combine IMUs with string-encoders. In this type of applications , the orientation is obtained by using an IMU, and the position of moving object is obtained using string encoders. The integration of IMU with string-encoders presents benefits encompassing computational efficiency, rapid responsiveness, environmental resilience, and higher level of accuracy.

In this thesis, a sensor system has been designed and manufactured, consisting of an IMU and six string-encoder position sensors, in addition to a robot arm, a spray system, and a computer. The computer is responsible for calculating the pose from sensor data, recording the results, and transferring this data to the robot arm.

The primary objective of this sensor system is to measure and record the three-dimensional position and orientation of a teaching tool—an operator-controlled spray gun equipped with sensors—during the process of spraying potassium permanganate onto a pair of jeans. The placement of string-encoders and used calculation techniques, which outlined in subsequent sections, distinguishes this study and contributes to superior results compared to similar studies (Won et al., 2009a, 2011; Jiafan et al., 2009; Andrade-Cetto & Thomas, 2006; Wang et al., 2017).

To mitigate issues associated with the inherent random walk and drift problems of IMUs, various types of sensor fusion techniques have been explored, and their orientation outputs are compared. Moreover, these results are compared with those obtained using two string-encoders for yaw angle calculation.

As an earning of this designed system, it is aimed to prevent workers from being harmed by exposure to chemicals during the spraying process.

As a result, this thesis practically demonstrated that the pose of an object in space

can be precisely determined by leveraging the power of sensor fusion through the combination of IMU and string encoders.

1.2 Literature Review

In the literature, numerous studies address sensor fusion, utilizing different sensor types, employing filter algorithms for instance Kalman filter, Particle filter, and Madgwick filter for accurate pose estimation.

Abyarjoo et al. (2015), developed and implemented a sensor fusion algorithm aimed at detecting orientation in three dimensions. The input to the fusion system comprises outputs from a gyroscope, an accelerometer and a magnetometer. They designed a Kalman filter to mitigate errors in inertial sensors by integrating data from gyroscope and accelerometer data. Moreover, they developed a tilt compensation unit to compute the heading of the system. The authors claimed that the experimental results prove the effectiveness of the proposed algorithm.

Won et al. (2008a), introduced an innovative approach to position and orientation estimation using a combination of Kalman filtering and particle filtering. The paper details the process of achieving highly accurate orientation and position estimates when equipped with one IMU and one position sensor. The authors conducted experiments comparing results without any filter, with a Kalman filter, and with their proposed filter. According to their findings, the proposed filtering technique effectively minimizes roll and pitch angle errors, although the yaw angle error exhibits a gradual increase over time, although at a significantly slower rate compared to the other two methods.

In their work, Guo et al. (2017) introduced a novel method that employs a quaternion-based Kalman filter scheme for orientation estimation, utilizing an MARG (magnetic, angular rate, and gravity) sensor array. The authors asserted that their innovative approach outpaces other Kalman-based methods in terms of speed and accuracy, and even surpasses the speed of certain complementary methods, all while

preserving attitude estimation accuracy.

Chow (2018), introduced an enhanced indoor pose tracking method that incorporates batch optimization and statistical sensor fusion, by using accelerometer, gyroscope and magnetometer. The authors asserted that their algorithm significantly enhanced in-situ dead-reckoning orientation accuracy, showing improvements ranging from 79.8% to 89.5%, and dead-reckoned positioning accuracy, demonstrating enhancements from 72.9% to 92.8%. Consequently, this led to a reduction in relative positioning error from the meter-level to the decimeter-level after ten seconds of integration, all achieved without making assumptions about the user's dynamics.

In their work, Meina et al. (2015) introduced a model fusion approach aimed at enhancing the effectiveness of Personal Dead Reckoning (PDR) Systems that utilize foot-mounted IMUs. Their solution involved estimating sensor orientation by combining Madgwick's algorithm with the Kalman filter. The authors asserted that their technique exhibited a 32% improvement in heading drift compensation for PDR.

Barraza-Madrigal et al. (2014), presented an algorithm to determine the instantaneous orientation of an object in 3D space. The orientation was determined using a Direct Cosine Matrix (DCM) achieved by combining three consecutive rotations around each of the main axes of the evaluated system, employing quaternions. The study's results demonstrated that the proposed algorithm, which includes PI controller feedback, not only eliminates drifting deviation but also reduces noise in both static positions and during movement. This allows for the accurate estimation of the orientation of an arbitrary object in 3D space.

Won et al. (2009c), presented an innovative methodology for pose estimation of an object using an IMU and a position sensor. The proposed method utilized a particle filter for orientation estimation and a Kalman filter for position and velocity estimation. Additionally, an expert system was employed to correct angular velocity bias measurement errors. The authors claimed that their proposed method significantly reduced orientation errors compared to errors obtained using only the EKF (Extended Kalman Filter) estimation method.

Won et al. (2009b), introduced a cost-effective tool tracking system employing an IMU and a magnetometer. The system computed the pose of a fastening tool based on the IMU data. Additionally, an expert system was implemented to identify the initial position, stationary state, and the fastened bolt. The authors claimed that their system effectively reduced both position and orientation errors, although it could not precisely determine the exact position and orientation.

Won et al. (2008b), introduced an expert system designed to identify the stationary state of an IMU. A Kalman filter was implemented to reduce sensor noises, enhancing the reliability of the system. Initial experimental results indicated that the proposed method effectively identified the stationary state and corrected tilt angle errors. In stationary conditions, the yaw angle remains unchanged, while the roll and pitch angles vary in relation to the average acceleration values. However, the second set of experimental results suggested that the proposed tilt angle correction method did not significantly improve the yaw angle when the IMU moved and returned to its original orientation.

In addition to these studies on sensor fusion algorithms, a substantial body of literature exists concerning movement recording systems that employ IMU and/or string encoders.

In the study by Jiafan et al. (2009), a new method for active, online, lead-through robot programming was explored. This method was based on a 6-DOF wire-based tracking device named RoboPuppet, featuring the 3-2-1 configuration. The kinematic analysis of this configuration was addressed by deriving analytic expressions for the unique solution of the pose of the RoboPuppet relative to the robot TCP (Tool Center Point) and by estimating the pose error resulting from dimensional deviations in the wire-based tracking device. The authors assert that the tracking errors of RoboPuppet in position and orientation are 5 mm and 6°, respectively. Figure 1.1 illustrates their designed system.

Andrade-Cetto & Thomas (2008), presented an active strategy for a wire tracking device. It has been demonstrated that by allowing the sensor platform to rotate about

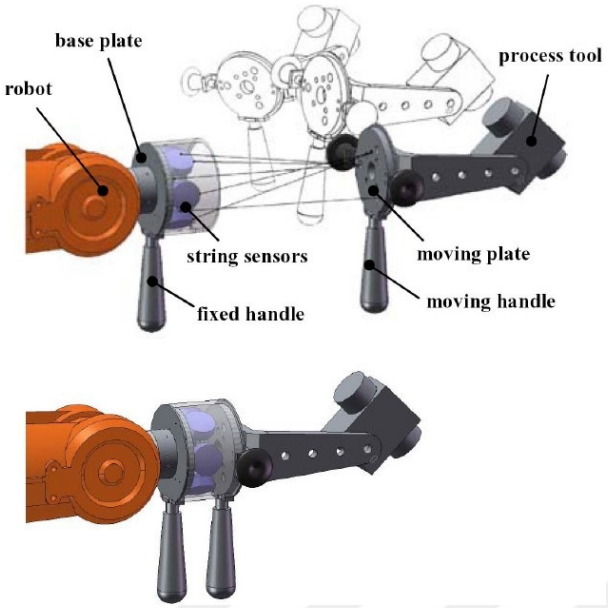


Figure 1.1 Designed system by (Jiafan et al., 2009)

its center, a broader range of motions can be tracked, reducing the required number of wires from six to three. They assert that this system also mitigates wire interference problems and minimizes the pulling force exerted by the device. Their results indicate that tracking errors of their system in position and orientation are at a maximum of 25 mm and 1.7°, respectively. Figure 1.2 illustrates their designed system.

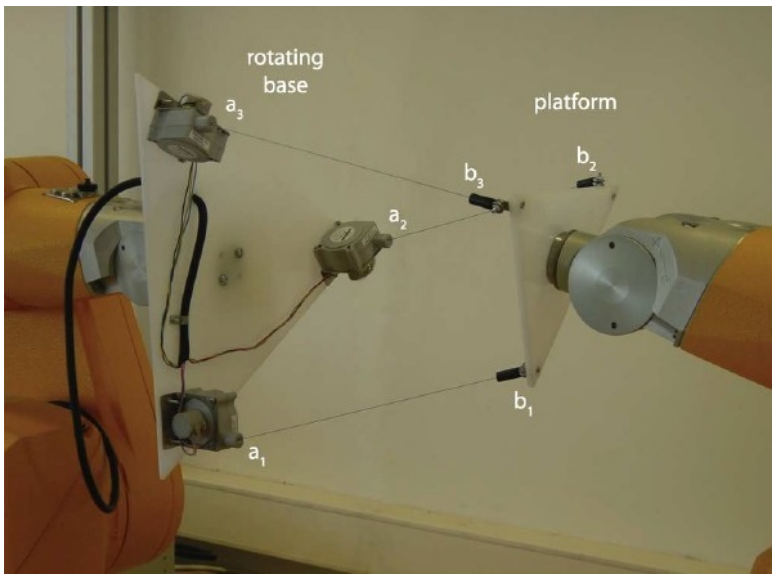


Figure 1.2 Designed system by (Andrade-Cetto & Thomas, 2008)

Thomas et al. (2005), addressed the challenge of estimating the pose of a moving rigid body by using six wire-based sensors attached to it. Their system utilized a trilateration formulation based on Cayley-Menger determinants. Experimental results revealed bias errors in their system attributed to wire-length inaccuracies. Figure 1.3 illustrates their designed system.

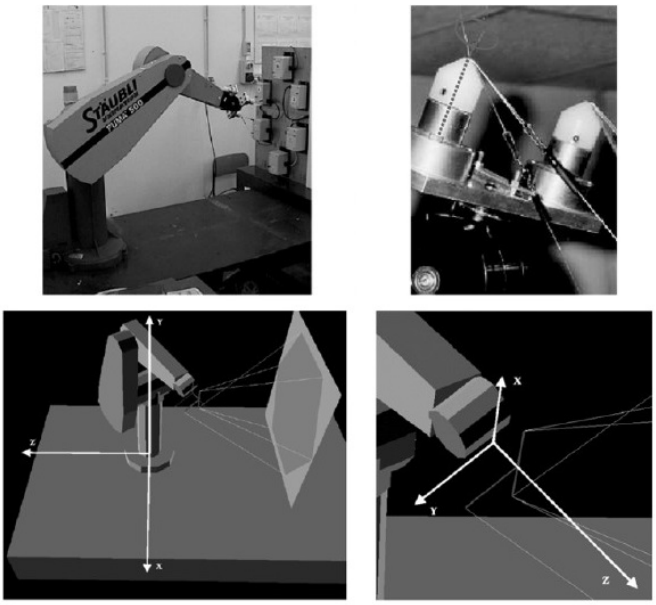
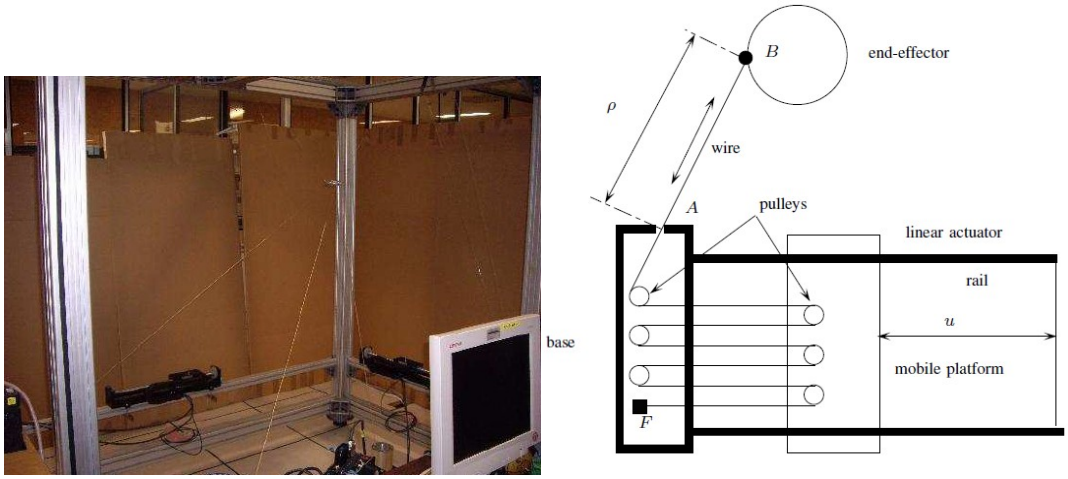


Figure 1.3 Designed system by (Thomas et al., 2005)

Merlet (2008), presented a robot utilizing a linear actuator and a pulley system, enabling increased modularity of the actuation system. The forward and inverse kinematics of this redundant system (comprising 7 wires) were discussed. While they introduced a solution for the forward kinematics problem, it was emphasized that this problem remains an open issue. Figure 1.4 illustrates their designed system.

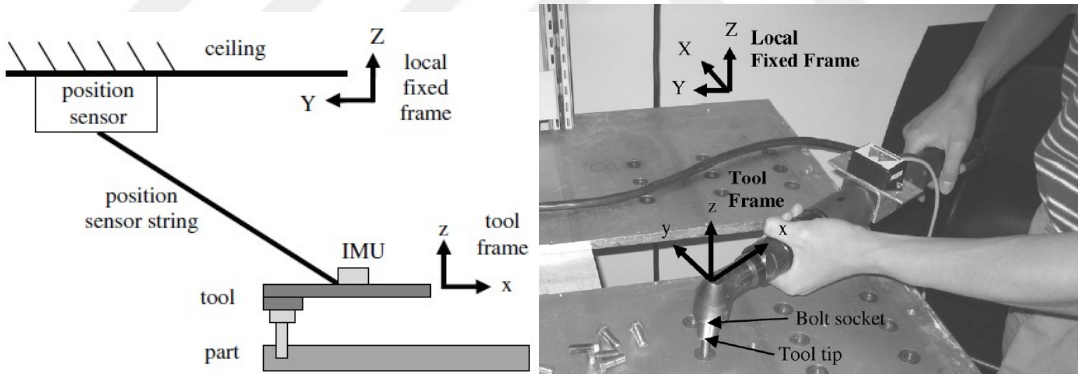
Won et al. (2009a), employed an intelligent system incorporating Kalman filters and a fuzzy expert system to track the tip of a fastening tool and identify the fastened bolt. Their system utilized one IMU and one position sensor to determine the orientation and center of mass location of the tool. While a Kalman filter was employed for orientation estimation, they noted an increase in orientation error over time due to the integration of angular velocity error. Consequently, they proposed a method to correct tilt angle and orientation errors using a fuzzy expert system. They asserted that their intelligent



(a) (b)

Figure 1.4 (a) The Marionet robot, (b) The wire system (Merlet, 2008)

system reduced the position error to a maximum of 15 mm. Figure 1.5 illustrates their designed system.



(a) (b)

Figure 1.5 (a) Illustration of the designed system, (b) Experimental Setup (Won et al., 2009a)

1.3 Organization of the Thesis

In Chapter 2, the early phases of this work and the designed experimental setup are presented, along with the theoretical background concerning the materials and methods

used in the experiments.

Chapter 3 presents experimental works on the pose estimation of the teaching tool. Additionally, studies conducted on ADIS16480 and MTi-3-DK are detailed.

In Chapter 4, experimental results on pose estimation are presented and discussed.

In Chapter 5, experimental works on sensor fusion algorithm designs are presented and outputs of these filters are compared with the real orientation data.

Chapter 6 concludes the thesis; the main contributions have been summarized, and future works have been discussed briefly.

CHAPTER TWO
MATERIALS AND METHODS

This chapter provides a detailed overview of the materials and methodologies utilized in this thesis. It covers the tools, equipment, and resources employed in the experimental setup. The section also outlines specific methodologies and techniques used for conducting the experiments.

2.1 Early Phases of the Work

In the early stages of the study, there is a need for an experimental setup to test the system for which theoretical calculations have been made. For this purpose, a test environment has been established, comprising a framework, string encoders, a teaching tool, ADIS16480 IMU, and a robot arm. Pant-shaped dummmies have been positioned across to the teaching tool and the robot arm. Two laser pointers are strategically positioned at the center of the teaching tool and the TCP of the robot arm. The designed test environment is shown in Figure 2.1.

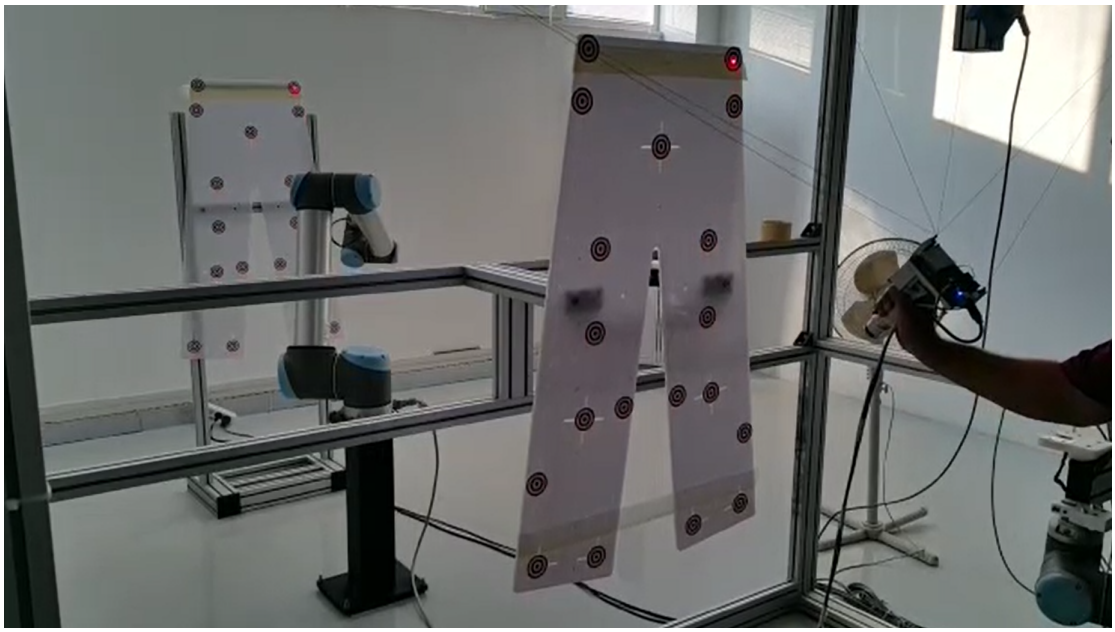


Figure 2.1 Designed Test Environment in the Early Phases of the Work

As illustrated in Figure 2.1, numerous markers have been positioned at the same

locations on both pant-shaped dummies. An individual carrying the teaching tool moves the laser pointer from one marker to another, and simultaneously, the robot replicates the same movement precisely.

Position calculations were conducted using four string encoders, and orientation angles were obtained from the EKF outputs of the IMU. However, it has been observed that the yaw angle output exhibits a drift problem over time in certain circumstances. Tests revealed that the yaw angle error can occasionally reach dramatic values during movement. Given that this occasional drift can impact the overall performance of the movement record system, two additional string encoders have been introduced and employed to calculate the yaw angle using the triangulation technique. At that time, this solution was the fastest and was implemented and tested within one day.

However, this drift issue prompted us to go deeply into the problem, initiating our efforts toward working on sensor fusion techniques.

2.2 Introduction of the Designed Experimental Setup

The designed experimental setup comprises two cabinet rooms made from chromium. The first cabinet is referred to as the movement record cabinet, while the second one is known as the spray application cabinet. They are built to the same size and positioned next to each other. A waterfall is designed to flow on the inner back faces of the cabinets to collect waste spray particles and divert them away from the operator's working area. Computer aided designs (cad) of the cabinets are shown in Figure 2.2.

As shown in the Figure 2.2 (a), the teaching tool, sensor system, and dummy legs are located in the movement record cabinet. An IMU is mounted on top of the teaching tool and six pieces of string-encoder position sensors are positioned at specific distances and directions relative to each other. A closer look to teaching tool and string-encoder placements are shown in Figure 2.3

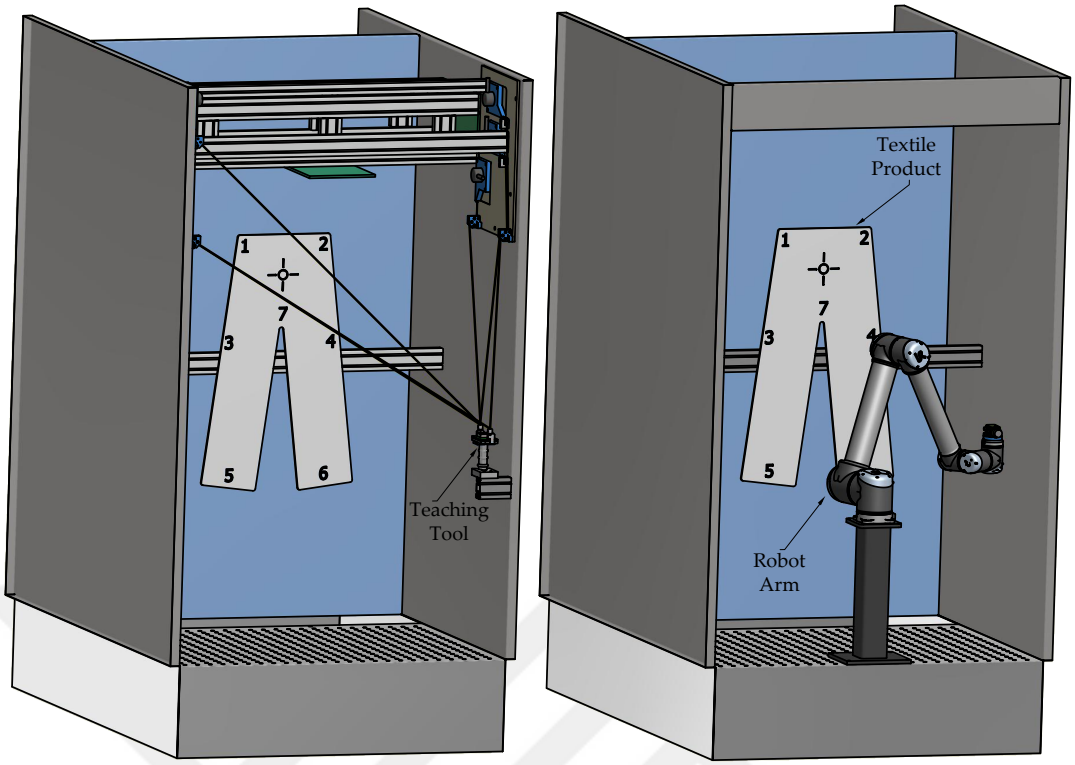


Figure 2.2 (a) Movement record cabinet, (b) Application cabinet

As shown in the Figure 2.3 (a), the strings from sensors labeled as S2,S3,S5 and S6 are connected to the junction point at the front side of the teaching tool, while strings from the sensors labeled as S1 and S4 are connected to the junction point at the rear side of the teaching tool.

The design of the teaching tool together with the IMU and its controller board, spray button and spraying nozzle is shown in the Figure 2.3 (b).

In movement record cabinet, the operator dresses the jean garment on dummy legs, inflates it with air, and to initiate the movement record operation, he takes the spray gun and performs the spraying process. The spray gun is equipped with a button for starting/stopping the spraying of Potassium Permanganate, and spraying timings are concurrently recorded with movements.

A robot arm and dummy legs are located in the application cabinet as shown in the

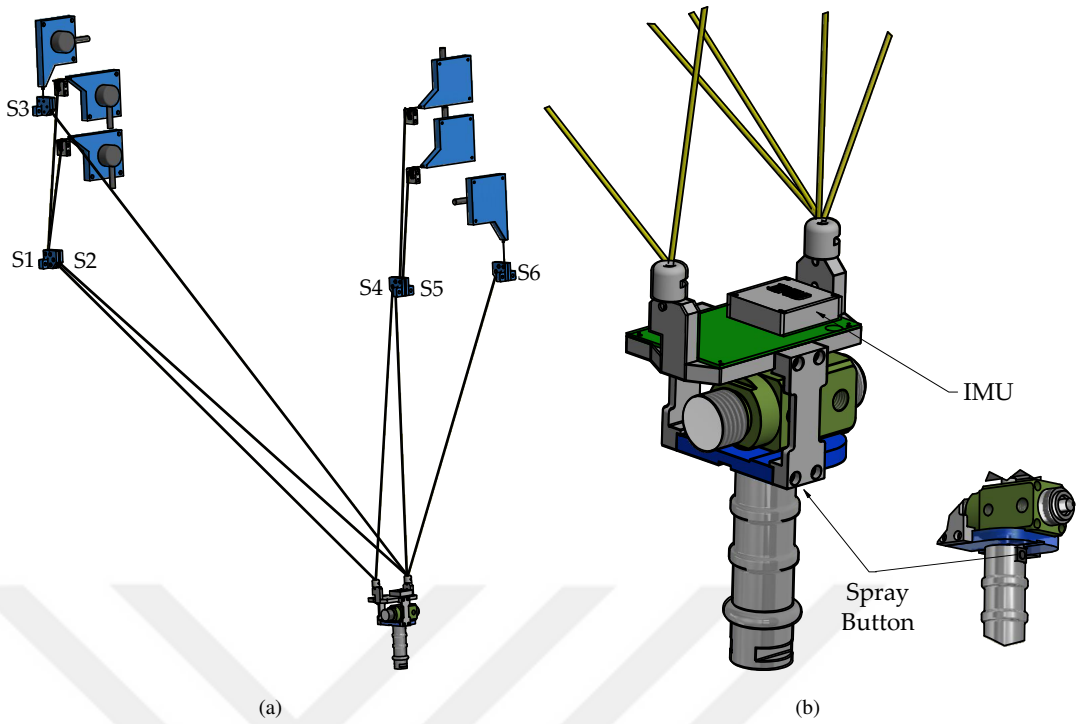


Figure 2.3 Closer look to the (a) String-encoder Placements , (b) Teaching Tool

Figure 2.2 (b). A nozzle for spraying purpose suitable to mount to the end-effector of the robot is designed and manufactured.

In addition, the experimental setup includes a designed spray system and a computer that executes the written program and designed user interface.

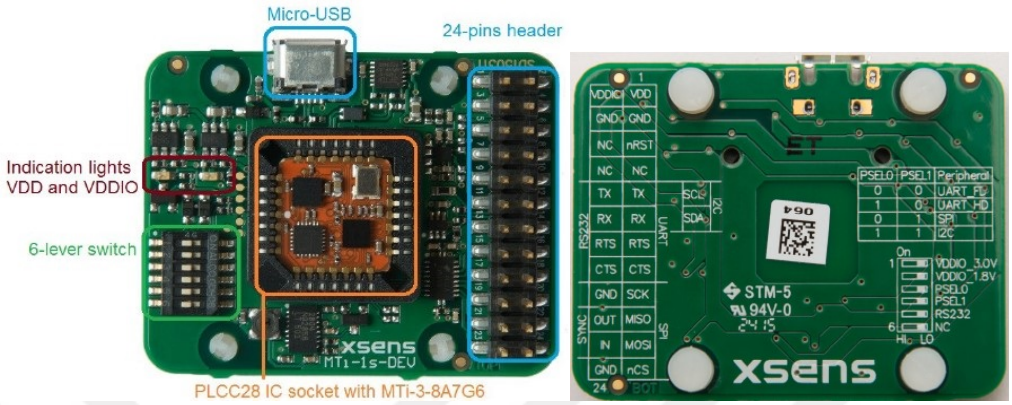
2.3 Sensors and Methods

2.3.1 IMU

Two different IMUs were used and tested during the experiments. Their test results is shown in the Chapter 2.

The first one is the MTi-3-DK produced by Xsens company, which comes with its own Development Kit design. This design eliminates the need for PCB design to

communicate and read IMU sensor data. The MTi-3-DK is equipped with an Arm Cortex M4 processor and an internal sensor fusion algorithm known as Xsens Kalman Filter (XKF3). Figure 2.4 provides top and bottom views of the MTi-3-DK.



(a) (b)

Figure 2.4 Xsens MTi-3-DK (a) Top view, (b) Bottom view

Important specifications of MTi-3-DK IMU are listed in Table 2.1.

Table 2.1 Important specifications of the Xsens MTi-3-DK IMU, Xsens Technologies B.V. (2016)

Property		**MTi-3-DK**		
In-run bias stability		6°/hr		
		Roll	Pitch	Yaw
Accuracy	Static	0.75°	0.75°	2°
	Dynamic	1°	1°	2°
		Gyro.(°/sec)	Accel.(g)	Magn.(mGs)
Dynamic Range		±2000	±16	±800
Noise Density		0.01	2×10^{-4}	0.2
Filter options		Xsens Kalman Filter (XKF3)		
Sample rate		Up to 1000 Hz		

The MTi-3-DK IMU is supported by the MT Manager software, developed by Xsens. This software enables real-time viewing of sensor data and 3D orientation outputs, allowing users to monitor and modify device settings. Figure 2.5 illustrates a sample data collection using the MT Manager software.

The MTi-3-DK also supports the MT Software Development Kit (SDK) and

Figure 2.5 Raw sensor data taken from MT Manager software

Low-Level Communication Protocol (LLCP). These features enable communication with the device and retrieval of raw sensor data from the IMU without the necessity of using the MT Manager software. The obtained raw sensor data is then utilized in the designed sensor fusion algorithms.

The second IMU tested and utilized in the experiments is the ADIS16480 produced by Analog Devices company. Similar to the MTi-3, this IMU features a gyroscope, accelerometer, and magnetometer, but it also incorporates an additional pressure sensor. The ADIS16480 provides a built-in adaptive EKF for accurate motion sensing and orientation estimation. The sensor is compatible with SDKs and communication protocols, facilitating easy access to raw sensor data for further processing.

Figure 2.6 provides visual of ADIS16480 IMU with its reference coordinate frame, and important specifications of the IMU are listed in Table 2.2.

These features collectively make the ADIS16480 IMU a versatile and reliable choice for applications that demand accurate motion and orientation sensing capabilities.

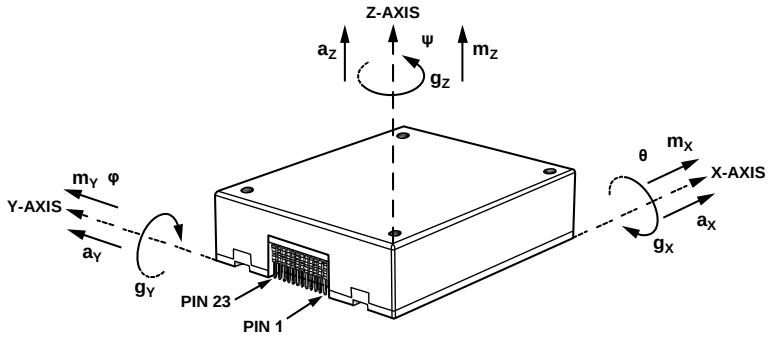


Figure 2.6 ADIS16480 IMU with its reference coordinate frame, Analog Devices Inc. (2017)

Table 2.2 Important specifications of the Analog Devices ADIS16480 IMU, Analog Devices Inc. (2017)

Property		**ADIS16480**		
Angular random walk		0.3°/ $\sqrt{hr}$		
Velocity random walk		0.029 m/sec/ $\sqrt{hr}$		
In-run bias stability		6°/hr		
		Roll	Pitch	Yaw
Accuracy	Static	0.1°	0.1°	0.3°
	Dynamic	0.3°	0.3°	0.5°
		Gyro.(°/sec)	Accel.(g)	Magn.(mGs)
Dynamic Range		±450	±10	±2500
Sensitivity/LSB		3.052×10^{-7}	1.221×10^{-8}	0.1
Noise Density		6.6×10^{-3}	6.7×10^{-7}	0.054
Filter options		Adaptive EKF		
Sample rate		700 - 2400 Hz		
Resolution		16-bit, two's complement format		

2.3.2 String-encoder Position Sensor

As previously mentioned, the proposed system incorporates six string-encoder position sensors, each consisting of two main components: an incremental encoder, SICK-DFS60B-S1AC04096, and a wire draw mechanism, SICK-PFG13-A1CM0544. The wire draw mechanism exhibits a measurement range from 0 to 5 m with a reproducibility of less than 0.5 mm. It can operate at a maximum speed of 4 m/s, with a maximum wire acceleration of 4 m/s^2. The incremental encoder generates 4096 pulses per revolution and offers a resolution of 0.09 mm. The sensor's maximum output frequency is 600 kHz. For a visual representation, refer to Figure 2.7, which

illustrates the string-encoder position sensor utilized in the designed system.

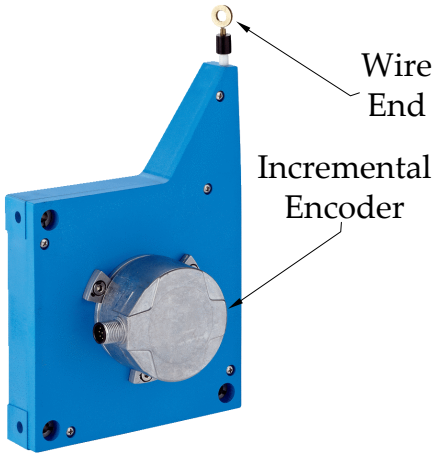


Figure 2.7 Wire draw mechanism and incremental encoder of string-encoder position sensor, Sick AG (2017)

The main bodies of the string-encoders are attached to the movement record cabinet, and individual wires pass through dedicated guidance components to avoid entanglement. The positions of the starting points of the wires are assumed to be at these guidance, making the positions of the main bodies of the string encoders unimportant.

The ring at the end of the wires are connected to the front or back side of the teaching tool, to form triangles between the wire starting points and the teaching tool, as illustrated in Figure 2.3 (a). These triangles are essential for measuring the 3D position and yaw angle, as explained in Chapter 3. Assuming the position of S1 as the center of the sensor coordinate system, Table 2.3 presents the locations of the wire starting positions.

Another advantage of string encoders is their assistance in bearing the weight of the teaching tool for the operator. As the bodies of the string encoders are positioned at elevated points compared to the teaching tool's operation site, the pulleys exert force to support the teaching tool. It's worth noting that this force might potentially impede the responsiveness of the teaching tool when operating at high speeds. Nevertheless, this aspect does not impact the proposed system design, as the system is intended for

Table 2.3 Positions of the wire starting points on the movement record cabinet in sensor coordinate system

Sensor Number	X pos. (mm)	Y pos. (mm)	Z pos. (mm)
S1	0	0	0
S2	0	0	0
S3	0	0	434
S4	0	1378	0
S5	0	1378	0
S6	-400	1378	0

tasks like spraying, which do not demand high speeds.

2.3.3 Robot Arm

The UR10, manufactured by Universal Robots, is employed in this study. It is an industrial robotic arm designed to offer flexibility and precision in a variety of applications. With a payload capacity of 10 kilograms, the UR10 can handle a range of tasks, making it suitable for diverse industries such as manufacturing, assembly, and material handling. This collaborative robot features a reach of 1300 millimeters, providing an extensive work envelope. It is a 6-DOF serial robot with six revolute joints. Each joint operates independently, driven by its own motor, providing a rotation range of ±360 degrees and a speed limit ranging from 120-180°/sec. Equipped with built-in force-torque sensors in each joint, the UR10 can easily collaborate with human operators, ensuring safety in shared workspaces. The robot boasts user-friendly programming through the UR Polyscope interface, allowing intuitive and efficient customization of tasks. Additionally, the UR10's repeatability of ±0.1 mm contribute to its reliability in performing intricate and repetitive tasks with high precision. The Universal Robot UR10 stands out as a versatile and collaborative robotic solution tailored to meet the demands of modern industrial automation.

The UR10 features an I/O port (GPIO) located inside the control box, which offers

both digital and analog input and output capabilities. Additionally, at the tool end of the robot, a tool I/O port is available, providing digital inputs and outputs along with analog inputs. These GPIO ports serve as interfaces for integrating various equipment, including sensors, motors, grippers, pneumatic valves, and safety devices, or for communication with external systems. The robot's end effector features an aluminum tool flange, facilitating seamless integration of tools or sensors. The UR10 has a maximum payload capacity of 10 kg, whereas the robot weight is 28.9 kg.

The UR10 offers three programming options at different levels: the Graphical User Interface Level (PolyScope GUI installed on the 12" touchscreen teach pendant that comes with the robot), the Script Level (URScript), and the C-API Level. URScript, a programming language developed by Universal Robots, features its own functions, flow control statements, data types, and variables tailored for monitoring and controlling the robot's movements and I/O ports. The execution of URScript programs occurs in real-time within the URControl Runtime Machine (RTMachine), and communication between URControl and the robot takes place at a frequency of 125 Hz.

Programming a UR10 robot at the Script Level involves creating a client application and establishing a connection to URControl using a TCP/IP socket Universal Robots (2016). Once the connection is set up, URScript programs can be transmitted from the computer, which runs the client application, to URControl as strings over the socket. The URControl executes essential calculations, including speeds, force and torque calculations, inverse kinematics, to precisely navigate the TCP of the robot to its designated target points. Subsequently, it generates and dispatches control commands to the motor drivers of the UR10.

The client application necessary for controlling the robot through Script Level programming is coded in the Pascal programming language, referred to in this thesis as the designed system program. This designed program creates a script program as an output by using the functions and variables of the URScript programming language, such as *movel*, *movej*, *get_inverse_kin* and *servoj*.

The script program contains crucial information, including the target position and target speed for the TCP of the robot. The designed program communicates with the URControl, transmitting commands at a 40 Hz sampling rate.

The Real-Time Data Exchange (RTDE) interface, developed by Universal Robots, is employed to gather real-time data from the robot, such as the position and orientation of the end-effector, actual speed and acceleration values, torques applied to motors and states of the GPIO port.

2.3.4 Spray System

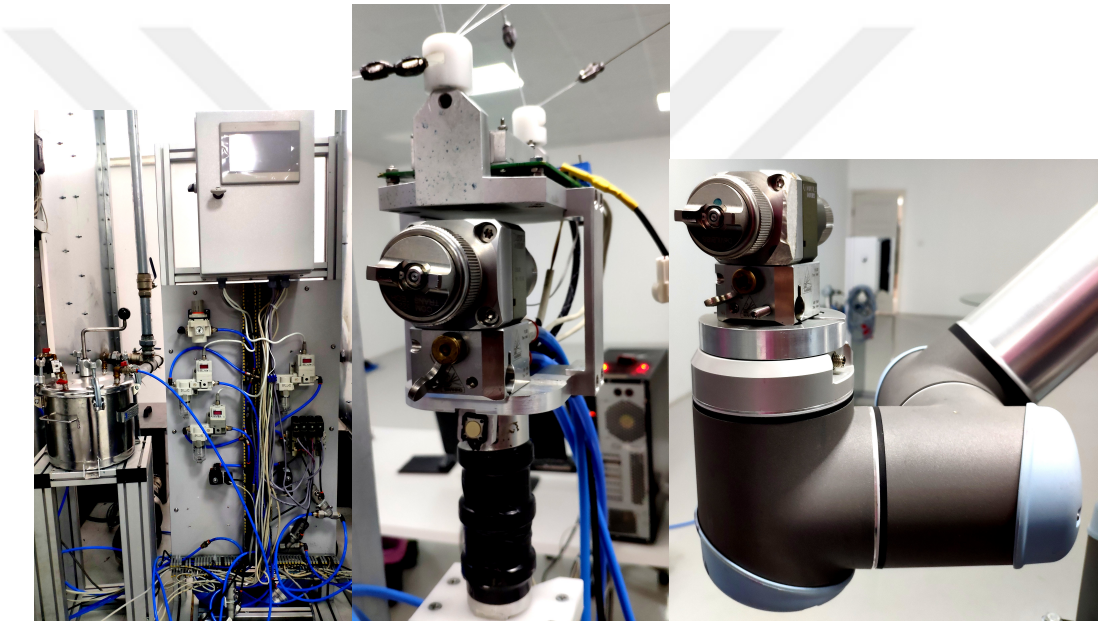
A spray system has been designed for use in both during the movement recording by an operator and the robot-executed spraying process. This system comprises various components, with the main components including two nozzle end-effectors (also known as spray-guns), a PLC, and an HMI. Auxiliary components covers mist separators, a dye reserve tank, one manual, and three electro-pneumatic air pressure regulators, relays, valves, and an electric panel box.

The nozzle end-effectors are mounted on the tool end of the UR10 robot and the teaching tool. The PLC unit and other electrical components are housed inside the panel box, while the HMI is mounted on the front side of the panel box. The PLC program is written in the Ladder Diagram (LD) programming language using Control FPWin Pro software developed by Panasonic Corporation. The GUI running in the HMI is designed using EasyBuilder Pro software from Weintek Labs. The PLC unit facilitates communication between the PC and HMI through the RS-232 protocol.

The reserve tank serves as a storage unit for the liquid dye solution. A mixer positioned atop the reserve tank effectively prevents residue accumulation. To ensure optimal performance, the manual air pressure regulator finely adjusts the compressed dry air to the required levels, facilitating supply to air-driven components. Mist separators play a crucial role in purifying the inlet compressed air by removing mist and particles. The outlets of these separators are connected to the electro-pneumatic

air pressure regulators. Notably, one electro-pneumatic regulator's outlet is connected to the reserve tank for dye pressure adjustment. Simultaneously, the outlets of the other two regulators are commonly connected to the nozzle-end effectors, allowing adjustments to atomizing air pressure and spray pattern size.

Electro-pneumatic regulators support the 4-20 mA analog communication protocol to read pressure setpoints from the PLC unit and send actual pressure values to the PLC unit. These parameters can be monitored from both the HMI of the spray system and the designed user interface running on the PC. Figure 2.8 illustrates the designed spray system with nozzle-end effectors located at the robot TCP and the teaching tool.

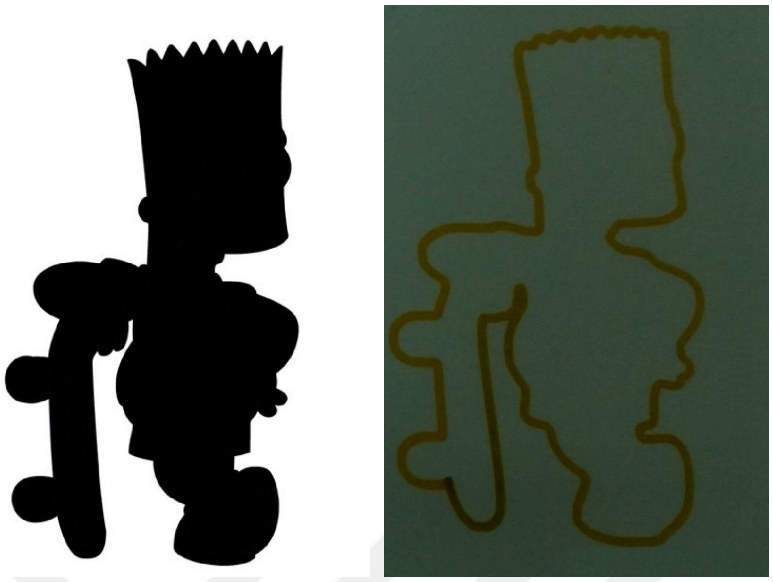


(a) (b) (c)

Figure 2.8 (a) Overview of the designed spray system, (b) Spray nozzle located on the teaching tool, (c) Spray nozzle mounted to robot TCP

A two-dimensional image measuring 450x618 pixels was employed to evaluate the precision of the spray nozzle system. The contours of the image were computed and translated into pose data. Subsequently, a paper was positioned in front of the robot, and the calculated pose data was transmitted to it. The entire spraying process took approximately 2 seconds for this evaluation. The outcome of the test is shown in Figure 2.9. Based on the results, the spray system demonstrated flawless performance,

meeting the requirements of the PP spraying applications.



(a) (b)

Figure 2.9 (a) Sample image, and (b) Spraying result of the nozzle system

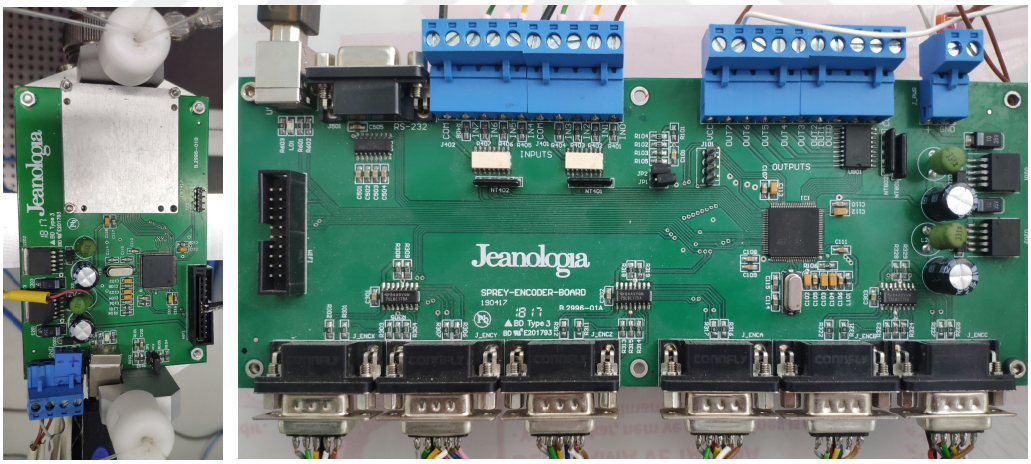
2.3.5 Controller Boards for Data Acquisition

Two controller boards, as depicted in Figure 2.10, have been designed and produced for the data acquisition purpose. Both controller boards have the same type of microcontroller, a 32-bit ARM Cortex-M4 core microcontroller (STM32F407 MCU), operating at 168 MHz speed. The microcontrollers' embedded programs, developed in the C++ programming language on Keil Microcontroller Development Kit (MDK).

The first controller board, denoted as the orientation board, is positioned atop the teaching tool. This board includes electrical components, mainly one ADIS16480 IMU, one microcontroller, and various circuit elements. The primary functionalities of the orientation board include supplying power to the ADIS16480 IMU, establishing communication with the IMU through the Serial Peripheral Interface (SPI) protocol, and enabling data transmission with the PC via the USB port. Notably,

the orientation board provides access to various outputs from the IMU, such as raw sensor data, orientation angle outputs, and timestamps. These outputs can be seamlessly accessed and retrieved from the PC through this board, offering a comprehensive interface for both data acquisition and adjustment of IMU settings.

Concurrently, the second controller board, denoted as the encoder board, is affixed to the right-hand side exterior surface of the movement record cabinet, and mainly equipped with a microcontroller, an I/O socket with 8 inputs and 8 outputs, six 8-pin female connectors, six buffer amplifiers, and a USB port. Signals from the string-encoder are transferred to controller board and undergo processing via the buffer amplifiers before being transmitted to the microcontroller. Subsequently, the microcontroller transmits these processed signals to the connected PC through the USB port, facilitating comprehensive data acquisition and analysis. Digital input and output ports are used to read the states of the spray button, and to control the spraying system.



(a) (b)

Figure 2.10 (a) Orientation board, placed atop the teaching tool, and (b) Encoder board, placed to the movement record cabinet

2.3.6 Master Program and User Interface

A PC, positioned between two cabinets, executes the master program developed in Pascal programming language using the Delphi integrated development environment (IDE). The data acquired from the controller boards is transmitted to the PC, where the master program captures these readings at a 40 Hz sampling rate. Simultaneously, it calculates and logs the pose information of the teaching tool. After the operator completes the movement record process, the pose logs are transformed into a URScript code using the built-in functions and variables of the URScript programming language. The resulting script code is then transmitted to the robot's controller, which stores it in its internal memory, offering the option to execute these recorded movements by the robot at any desired moment.

A user interface has been designed to enable the monitoring and adjustment of variables, including the initialization parameters of the string-encoders, parameter selections for the IMU, real-time pose information display, the ability to show raw sensor data, and initiate/terminate the movement recording process. The main page of the user interface is shown in the Figure 2.11.

2.3.7 Position Calculation Techniques

In the studies on geometry and geospatial positioning, various techniques are employed to determine the location of points in space. The Law of Cosines, a fundamental principle in trigonometry, plays a crucial role in solving triangles when the lengths of two sides and the included angle are given. It states that the square of one side of a triangle is equal to the sum of the squares of the other two sides, accounting for the cosine of the included angle. This law finds applications in scenarios where distance measurements and angles are utilized, providing a powerful tool for geometric calculations.

Trilateration, on the other hand, is a technique widely used in positioning systems to locate a point in space based on its distance from three known points. By

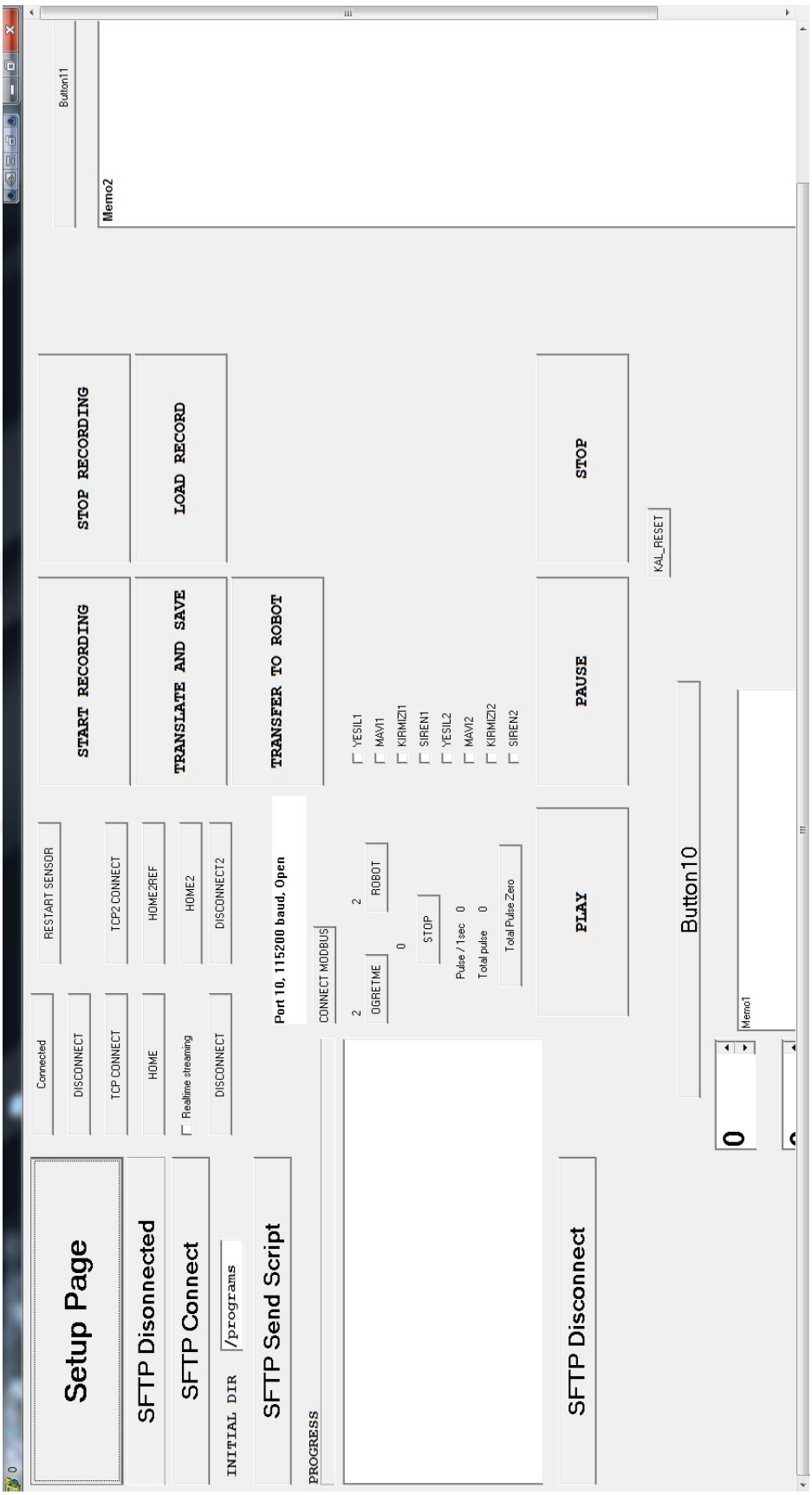


Figure 2.11 User interface runs on the PC

measuring the distances from the unknown point to each of the reference points, trilateration involves creating circles (or spheres in three dimensions) centered at these reference points with radii corresponding to the measured distances. The intersection of these circles determines the possible locations of the unknown point. Trilateration is commonly employed in GPS systems, where satellites serve as the reference points, and the distances are determined through signal travel time.

Multilateration extends the concept of trilateration to involve measurements from more than three reference points. In scenarios where additional reference points are available, multilateration provides enhanced accuracy and redundancy. It is particularly valuable in environments where line-of-sight measurements may be obstructed, allowing for more robust positioning solutions. Aircraft surveillance systems often leverage multilateration to precisely determine the location of aircraft based on the time difference of arrival of signals from multiple ground-based transmitters.

In summary, the Law of Cosines, Trilateration, and Multilateration are powerful techniques employed in diverse fields for spatial positioning and location determination. Each method offers unique advantages and is applied based on the specific requirements of the given application, whether it's in geometry, navigation, or surveillance systems.

2.3.8 Sensor Fusion and Filtering Techniques

Sensor fusion plays a pivotal role in integrating data from multiple sensors to enhance the accuracy and reliability of measurements, particularly in the context of geometry and geospatial positioning. Sensor filtering methods are employed to reduce noise, enhance signal quality, and improve the accuracy of measurements, ensuring that the data used for further analysis or decision-making is reliable. Figure 2.12 shows how sensor fusion and filtering is works.

This section provides an overview of prominent sensor fusion and filtering

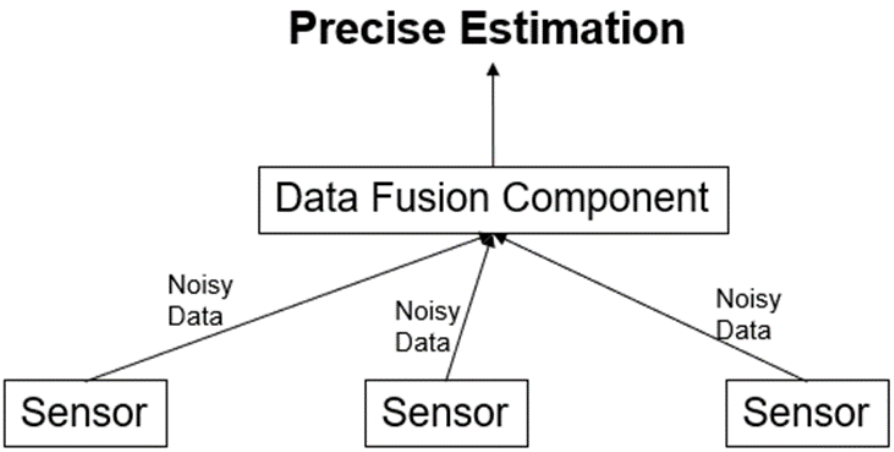


Figure 2.12 Visualization of Sensor Fusion and Filtering

techniques in the literature.

2.3.8.1 Kalman Filter

The Kalman Filter is a recursive algorithm employed for the estimation of the state of a dynamic system. It functions by processing a sequence of measurements over time, integrating the predictions derived from a dynamic model with the actual measurements.

The Kalman Filter involves two main steps: prediction and update. The state of the system at time k, denoted as x_k, is estimated based on the predicted state from the previous time step ($\hat{x}_{k-1}$) and the system dynamics represented by the state transition matrix (A).

Prediction Step:

$$\hat{x}_k = A \cdot x_{k-1} \tag{2.1}$$

$$P_k = A \cdot P_{k-1} \cdot A^T + Q \tag{2.2}$$

In these equations, P_k is the error covariance matrix, and Q represents the process noise covariance.

The prediction step calculates the estimated state ($\hat{x}_k$) and updates the error covariance matrix (P_k) based on the system dynamics and process noise.

Update Step:

$$K_k = P_k \cdot H^T \cdot (H \cdot P_k \cdot H^T + R)^{-1} \tag{2.3}$$

$$x_k = \hat{x}_k + K_k \cdot (z_k - H \cdot \hat{x}_k) \tag{2.4}$$

$$P_k = (I - K_k \cdot H) \cdot P_k \tag{2.5}$$

In these equations, K_k is the Kalman Gain, H is the observation matrix, R represents the measurement noise covariance, and z_k is the measurement vector.

The update step refines the estimated state (x_k) using the Kalman Gain, which is adjusted based on the error covariance, observation matrix, and measurement noise.

In conclusion, the Kalman Filter excels in providing an optimal estimation framework for linear systems, making it a versatile tool in applications where accurate state estimation is crucial. Its ability to handle noisy measurements and dynamically adjust predictions based on new information makes it a fundamental component in real-time systems and control applications.

2.3.8.2 Extended Kalman Filter

The Extended Kalman Filter represents an extension of the conventional Kalman filter specifically devised to address non-linear systems. In contrast to the original Kalman filter, which presupposes linear relationships among variables, the EKF manages non-linearities by linearizing the system at each time step. This makes it suitable for a broader range of applications where the system dynamics exhibit non-linear characteristics.

The core equations of the Extended Kalman Filter are derived from the same principles as the Kalman filter but involve the Jacobian matrix to linearize the non-linear functions. The key steps in the EKF algorithm include prediction and update, similar to the Kalman filter:

Prediction Step:

$$\hat{x}_k = f(\hat{x}_{k-1}, u_k) \tag{2.6}$$

$$P_k = A \cdot P_{k-1} \cdot A^T + Q \tag{2.7}$$

Here, f represents the nonlinear state transition function, u_k is the control input, and A is the Jacobian matrix of f evaluated at $\hat{x}_{k-1}$.

Update Step:

$$K_k = P_k \cdot H^T \cdot (H \cdot P_k \cdot H^T + R)^{-1} \tag{2.8}$$

$$x_k = \hat{x}_k + K_k \cdot (z_k - h(\hat{x}_k)) \tag{2.9}$$

$$P_k = (I - K_k \cdot H) \cdot P_k \tag{2.10}$$

In the update step, h represents the measurement function, and H is the Jacobian matrix of h evaluated at $\hat{x}_k$. The measurement vector is denoted by z_k.

The EKF linearizes the system at each time step using the Jacobian matrices, allowing it to handle non-linear dynamics. However, it is essential to note that the accuracy of the EKF depends on the accuracy of the linearization, and it may encounter challenges in highly non-linear systems.

In summary, the Extended Kalman Filter is a powerful tool for estimating the state of non-linear systems by iteratively predicting and updating the state using linearized models.

2.3.8.3 Complementary Filter

Unlike the Kalman Filter and Extended Kalman Filter, the Complementary Filter is a simpler alternative that doesn't involve complex matrix calculations. A Complementary Filter combines low-pass and high-pass filters and it is commonly used for fusing accelerometer and gyroscope data to obtain more accurate orientation estimates. The mathematical background involves blending the outputs of the two filters based on predefined weighting factors.

$$\theta_{complementary} = \alpha \times \theta_{Acc} + (1 - \alpha) \times \theta_{Gyro} \tag{2.11}$$

where $\theta_{complementary}$ stands for the final orientation estimation, θ_{Acc} signifies the angle computed from accelerometer data, θ_{Gyro} signifies the angle computed from gyroscope data, and α is a tuning parameter between 0 and 1 that determines the contribution of each sensor.

2.3.8.4 Particle Filter

The Particle Filter, also known as the Sequential Monte Carlo (SMC) method, is a probabilistic Bayesian filter used for state estimation in nonlinear and non-Gaussian systems. They utilize a set of particles to represent the system's possible states and update their weights based on measurements.

Following equations collectively describe the Particle Filter's sequential process of predicting the state based on the motion model, updating the weights based on observed measurements, normalizing the weights, and resampling to ensure an accurate representation of the system's state.

Prediction Step:

$$x_k^{[i]} \sim p(x_k|u_k, x_{k-1}^{[i]}) \tag{2.12}$$

The prediction step involves sampling particles from the motion model, where u_k is the control input.

Update Step:

$$w_k^{[i]} = p(z_k|x_k^{[i]}) \tag{2.13}$$

In the update step, the weights are computed based on the likelihood of the observed measurement z_k.

$$w_k^{[i]} = \frac{w_k^{[i]}}{\sum_j w_k^{[j]}} \tag{2.14}$$

This step normalizes the weights to ensure they sum up to 1.

$x_k^{[i]}$ is resampled with probability proportional to $w_k^{[i]}$

The resampling step ensures that particles representing more probable states are replicated, while less probable ones are discarded.

2.3.8.5 Madgwick Filter

The Madgwick Filter is specifically designed for orientation estimation using data from IMUs, by Madgwick et al. (2011). It operates on quaternion representations of orientation and employs gradient descent optimization. The mathematical background involve quaternion algebra and sensor fusion principles.

Orientation Update:

$$\dot{q} = \frac{1}{2} q \otimes \begin{bmatrix} 0 \\ \omega \end{bmatrix} - \beta \cdot S(q) \cdot (a - q \otimes (q \otimes a \otimes q)^*) \tag{2.15}$$

In this equation: - q represents the quaternion. - ω is the angular rate vector from the gyroscope. - β is a filter parameter. - $S(q)$ is a function that transforms the accelerometer measurements into the quaternion space.

Quaternion Update:

$$q_{\text{est}} = q_{\text{est}} + \dot{q} \cdot \text{sample\_period} \tag{2.16}$$

This equation updates the estimated quaternion based on the quaternion rate of change.

Normalization:

$$q_{\text{est}} = \frac{q_{\text{est}}}{|q_{\text{est}}|} \tag{2.17}$$

Normalization is performed to ensure that the quaternion remains a unit quaternion.

2.3.8.6 MATLAB Sensor Fusion and Tracking Toolbox

MATLAB introduced a new toolbox in its R2018b version, known as the Sensor Fusion and Tracking Toolbox. This toolbox incorporates algorithms and tools for designing, simulating, and analyzing systems that fuse data from various sensors to estimate position, orientation, and situational awareness. The reference examples provided within the toolbox serve as foundational starting points for implementing various components in airborne, ground-based, shipborne, and underwater surveillance, navigation, and autonomous systems.

One of its significant features is the inclusion of various multi-object trackers, sensor fusion filters, motion and sensor models, and data association algorithms. These components collectively empower users to design and implement resilient sensor fusion architectures.

A key aspect of the toolbox is its capability to evaluate fusion architectures using both real-world and synthetic data. This feature is pivotal for assessing the performance and accuracy of developed systems across different scenarios.

The Sensor Fusion and Tracking Toolbox supports a wide array of sensors, including RF (Radio Frequency), acoustic, EO/IR (Electro-Optical/Infrared), and GPS/IMU. This flexibility allows users to work seamlessly with diverse sensor types commonly encountered in surveillance, navigation, and autonomous systems.

In summary, MATLAB's Sensor Fusion and Tracking Toolbox serves as a comprehensive and versatile solution for researchers, engineers, and developers working on projects involving sensor fusion.

CHAPTER THREE
EXPERIMENTAL WORKS ON POSE ESTIMATON

This chapter comprehensively describes the detailed implementation of materials and methodologies, as outlined in Chapter 2, to ensure the successful execution of the experiments.

3.1 Position Estimation of the Teaching Tool

The string-encoder position sensors employed in the experimental setup utilize incremental encoders that start counting from zero by their inherent nature. To facilitate their accurate functioning, it is necessary to initialize these sensors at a known position where all wire lengths are known. Therefore, a bracket is manufactured to position the teaching tool, and the corresponding wire lengths are measured and input into the master program. This designated position is referred to as the home position, serving as the central reference point within the global coordinate system. When the operator starts the movement recording process by clicking the *start recording* button on the User Interface, the initialization process is completed in the beginning, followed by the calculations of x, y, and z positions are started within the master program.

The process of choosing the positions of the guidance components and measuring the positions of the tip of the teaching tool is elucidated as follows:

1. The starting point, denoted as S2, is selected as the center of the global coordinate system. The second point must be chosen at a distinct location along the observed axis, maintaining the same positions on the other two axes. Assuming the observed axis is the Z axis, the position (0, 0, 434), denoted as S3, is selected as the second point. These designated points are where the guidance components are affixed.

2. The wires from these two anchor points are attached to the front junction point of the teaching tool.

3. In the given scenario, where a triangle is formed with all three side lengths known, the distances between two anchor points are fixed, and the distances from the anchor points to the teaching tool are measured using string encoders, the Law of Cosines can be applied to this triangle. The Law of Cosines allows for the calculation of angles or side lengths in a triangle when the lengths of all three sides are known.

4. Determine the observed axis position of the teaching tool using the formulas provided. The calculation procedure for the x position is depicted in Figure 3.1.

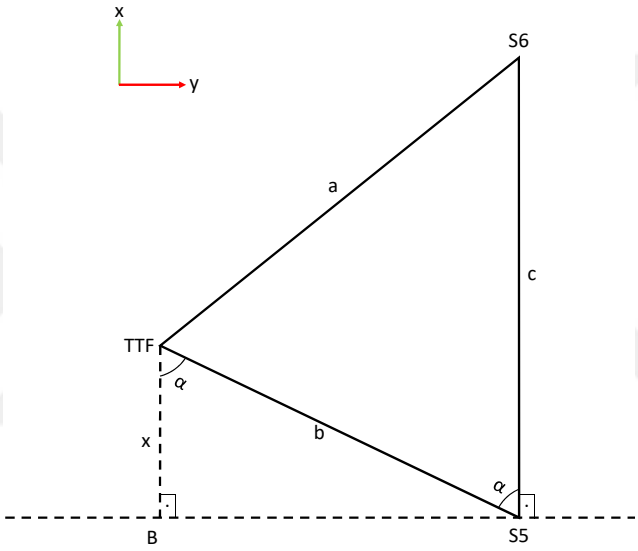


Figure 3.1 The triangle to calculate the x-axis position

The law of cosines states that;

$$\gamma = \arccos\left(\frac{b^2 + c^2 - a^2}{2bc}\right) \tag{3.1}$$

In the triangle illustrated in Figure 3.1, TTF represents the teaching tool front junction point of the four string-encoders. The length to be calculated is denoted as x. The length c corresponds to the fixed distance between the guidance components of the string-encoders S5 and S6, measuring 400 mm. The lengths a and b are obtained

from the readings of the S5 and S6 string-encoders. Applying the law of cosines through the formula in Equation (3.2) to the triangle S5TTFS6 allows the calculation of cosα. Subsequently, the x length in the triangle TTFS5B can be determined using the following equation:

$$x = \cos\alpha * b \tag{3.2}$$

Similar to x-axis position calculation, sensors S2 and S5 is used to create the triangle S2TTFS5 to calculate y-axis position as illustrated in Figure 3.2. The length to be calculated is denoted as y where the length c corresponds to the fixed distance between the guidance components of the string-encoders S2 and S5, measuring 1378 mm. The lengths a and b are obtained from the readings of the S5 and S2 string-encoders, respectively. Applying the law of cosines through the formula in Equation (3.1) to the triangle S2TTFS5 allows the calculation of cosα. Subsequently, the y length in the triangle TTFBS2 can be determined using the Equation (3.2).

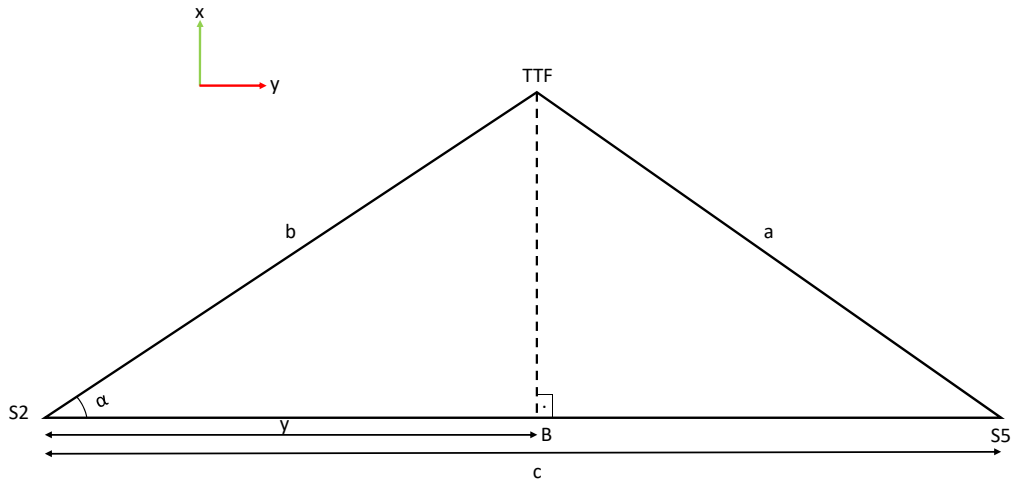


Figure 3.2 The triangle to calculate the y-axis position

And finally, sensors S2 and S3 is used to create the triangle S2TTFS3 to calculate z-axis position as shown in Figure 3.3. The length to be calculated is denoted as z

where the length c corresponds to the fixed distance between the guidance components of the string-encoders S2 and S3, measuring 434 mm. The lengths a and b are obtained from the readings of the S3 and S2 string-encoders, respectively. Applying the law of cosines through the formula in Equation (3.1) to the triangle S2TTFS3 allows the calculation of cosα. Subsequently, the z length in the triangle TTFBS2 can be determined using the Equation (3.2).

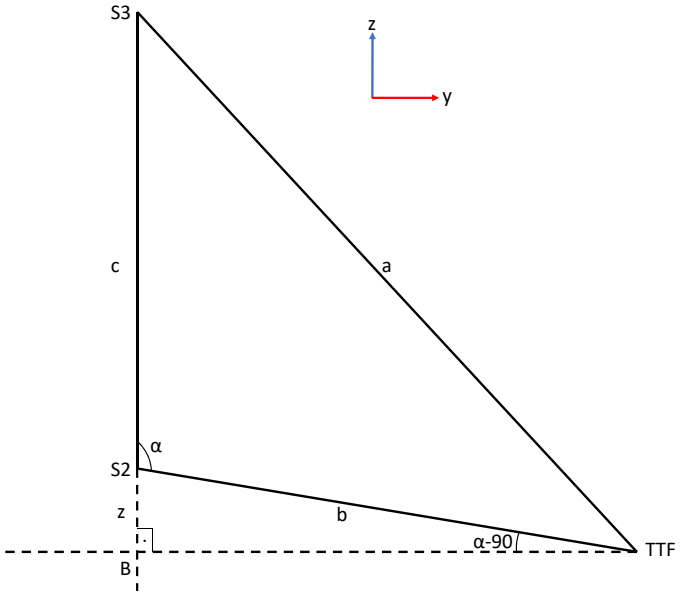


Figure 3.3 The triangle to calculate the z-axis position

3.2 Orientation Estimation of the Teaching Tool

The orientation angles of the teaching tool are directly obtained from the ADIS16480 IMU at a sampling rate of 40 Hz. The internal EKF of the ADIS16480 is activated to enhance orientation estimations. However, it has been noted that the yaw angle output of the EKF exhibits noticeably slow response times when the IMU's magnetometer is enabled. To address this issue, the magnetometer is disabled. It has also come to our attention that the yaw angle output of the EKF in the IMU demonstrates relatively high errors in time.

When the teaching tool is stationary, IMU sensor readings must remain constant. Accelerometer sensor readings should be 0, 0, 1g relative to the x-y-z axes, and gyroscope readings on all axes should be zero. However, as explained before, achieving perfect constancy is challenging due to their nature. In particular, the yaw angle is very noisy, because of the z-axis is parallel to the gravity vector. Roll and pitch angle measurements of the IMU consistently exhibit better accuracy than the yaw angle measurement.

According to the data presented in Table 2.2, the ADIS16480 is specified to have a dynamic accuracy of 0.3° for roll and pitch angles, and 0.5° for yaw angle. However, actual test results revealed a dynamic accuracy of approximately 1° for roll and pitch angles, with less precise outcomes for the yaw angle. Notably, dynamic movement tests on the ADIS16480 demonstrated that the yaw angle error could increase up to 10° within 30 seconds of movement. This error tends to escalate rapidly even when the ADIS16480 is stationary. The test results, illustrating the performance of the ADIS16480 in both dynamic movement and stationary states, are depicted in Figure 3.4.

The data in Figure 3.4 shows that the ADIS16480 exhibits significant attitude drifts in its outputs, particularly in the yaw angle output. The yaw angle output shows issues with random walk and bias stability. Due to these reasons, the roll and pitch angle outputs of the ADIS16480 are used, while the yaw angle output is not used for the orientation estimation of the teaching tool.

To enhance the overall system accuracy, the yaw angle is measured using string-encoder position sensors, as illustrated in Figure 3.5. From now on, the yaw angle calculated by using string-encoders will be denoted as *Calculated Yaw*, while the yaw angle output of the IMU will be denoted as *IMU yaw*.

In Figure 3.5, TTF represents the teaching tool front junction point of four string-encoders, while TTB denotes the teaching tool back junction point of two string-encoders. The y-position of TTF is measured using S2 and S5 string-encoders, while the y-position of TTB is measured using S1 and S4 string-encoders. The

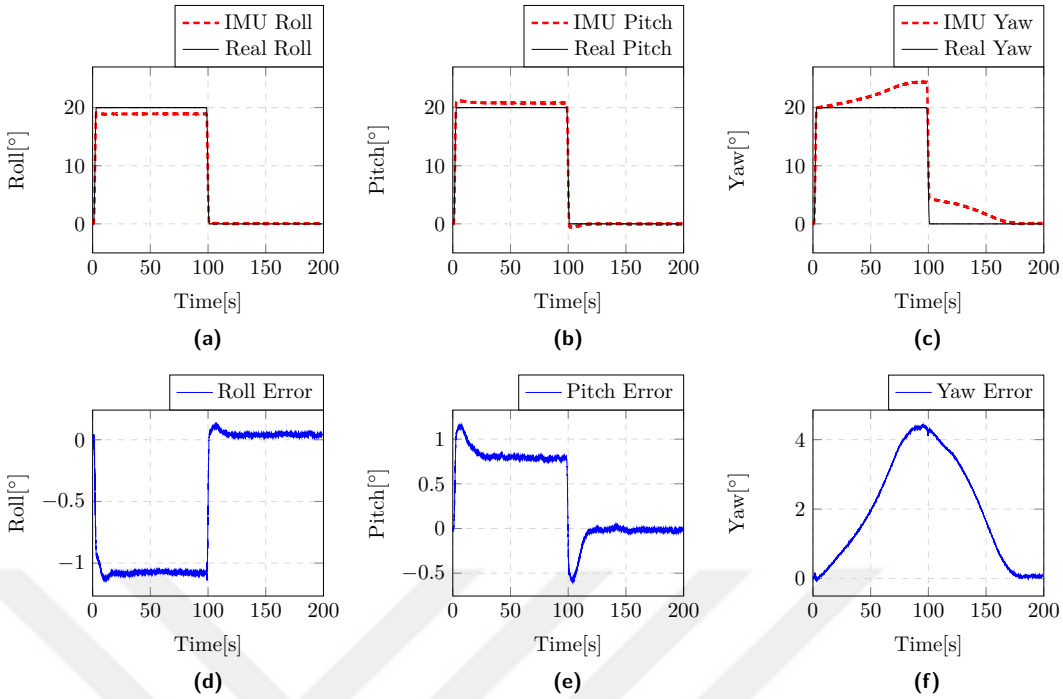


Figure 3.4 Test of the Euler angle outputs of the ADIS16480 on dynamic movement and stationary state (a) Roll angle, (b) Pitch angle, (c) Yaw angle, (d) Roll angle error, (e) Pitch angle error, and (f) Yaw angle error

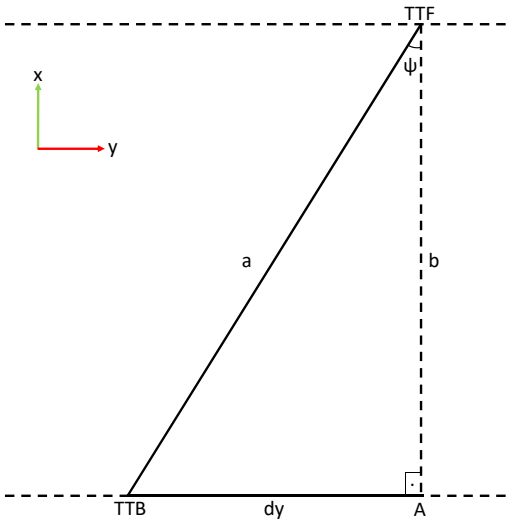


Figure 3.5 The triangle to calculate the yaw angle

difference between these two measurements, denoted as *dy*, is calculated. The fixed distance *a* has a length of 123.5 mm. The yaw angle can be determined using the following equation:

$$\psi = \arcsin\left(\frac{dy}{a * \cos\beta}\right) * \frac{180}{\pi} \tag{3.3}$$

where β represents the pitch angle of the teaching tool, obtained directly from the IMU readings.

Unlike the yaw angle output of the IMU EKF, this alternative solution offers stability and accuracy in its results, free from the presence of noise or the accumulation of errors.

3.3 Interpreting Raw IMU Data from ADIS16480

The error in the yaw angle output of the IMU EKF directs us to focus on the development and implementation of sensor fusion techniques, aiming for improved and more accurate results. To achieve this, capturing raw IMU sensor data is necessary. The ADIS16480 IMU provides various raw data outputs, including timestamp, gyroscope, accelerometer, magnetometer, barometer, temperature, delta angle, quaternion, rotation matrices, etc. Considering all the data that may be required from the raw data list provided by the IMU, 44 data types were selected and sampled at a frequency of 100 Hz.

ADIS16480 IMU provides all these data in hexadecimal (HEX) format. To calculate Euler Angles from raw IMU sensor data, it is essential to convert these HEX values to physical units following the explanations in the IMU's user manual. To illustrate this interpretation process, we will present the calculation of X-axis gyroscope data from raw sensor data here.

The raw data from the X-axis gyroscope is presented by the IMU in two words: a low word and a high word. The high word represents the most significant bits, while the low word represents the least significant bits, as illustrated in Figure 3.6

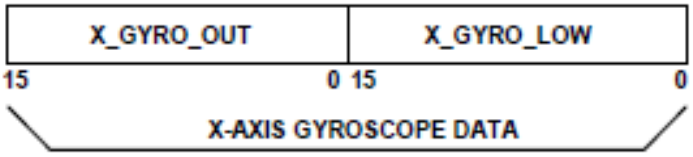


Figure 3.6 Contributions of most significant and least significant words to gyroscope X-axis output calculation

The weights of the bits of the most significant word are presented in Table 3.1. Each bit carries equal weight, which is 0.02 °/ sec. The Most Significant Bit (MSB) in the least significant word has a weight of 0.01 °/ sec, and each subsequent bit has half the weight of the previous one.

Table 3.1 X_GYRO_OUT Register (Page 0, Base Address = 0x12), Analog Devices Inc. (2017)

Bits	Description
[15:0]	X-axis gyroscope data; twos complement, 0°/sec = 0x0000, 1 LSB = 0.02°/sec

A MATLAB code script, which has been written to convert the raw data from accelerometer, gyroscope and magnetometer sensors to physical units based on the provided information on the datasheet of the ADIS16480 IMU is shown in Figure 3.7.

3.4 Studies with Xsens MTi-3-DK

Studies on sensor fusion techniques with the Xsens MTi-3-DK IMU carried on by using Xsens Device API (XDA) Library. The XDA Library is a SDK provided by Xsens, and it provides a set of functions and tools that enable communication with Xsens devices, allowing to access motion data, sensor information, and other features.

By using features of this library, a MATLAB function was written to communicate and capture real time sensor data from the MTi-3 IMU. To call this MATLAB function and draw the corresponding graphs using the values returned by the function, a main program has been written in MATLAB.

```
%% Gyroscope X Axis Calculation

for i = 1: row
X_Gyro_Low_Byte = 0;
X_Gyro_Out_Hex = data.kayit.X_Gyro_Out(i);
X_Gyro_Out_Dec = hex2dec(X_Gyro_Out_Hex);
X_Gyro_Out_Dec = twos2decimal(X_Gyro_Out_Dec,16);

X_Gyro_Low_Hex = data.kayit.X_Gyro_Low(i);
X_Gyro_Low_Dec = hex2dec(X_Gyro_Low_Hex);
X_Gyro_Low_Dec = twos2decimal(X_Gyro_Low_Dec,16);

if(X_Gyro_Out_Dec<0)
X_Gyro_Out_Dec = X_Gyro_Out_Dec*-1;
Signal_out = -1;
else
Signal_out = 1;
end

if(X_Gyro_Low_Dec<0)
X_Gyro_Low_Dec = X_Gyro_Low_Dec*-1;
Signal_low = -1;
else
Signal_low = 1;
end

X_Gyro_Low_Bin = de2bi(X_Gyro_Low_Dec,16);

for j = 1:16
X_Gyro_Low_Byte = X_Gyro_Low_Byte + X_Gyro_Low_Bin(j)*low_weight_gyrobit(j);
end

X_Gyro_Out_Byte = X_Gyro_Out_Dec * Out_Weight_Gyro * Signal_out;

G_x(i) = X_Gyro_Out_Byte + (X_Gyro_Low_Byte * Signal_low);

end

G_x = G_x'; % unit is deg/sec
Gx_rad = G_x * pi / 180.0; % unit is rad/sec
```

Figure 3.7 MATLAB Code for raw data interpretation from gyroscope x axis readings

The graphs of the sample data, which were obtained using the written MATLAB code and function, are presented in Figure 3.8. The results from many conducted tests with this IMU indicate that the accuracy of the IMU is insufficient to achieve the goals of the work, which is to obtain accurate Euler angle values during movement recording process. Thus, a different IMU model with better specifications, when compared their catalog informations, has been employed.

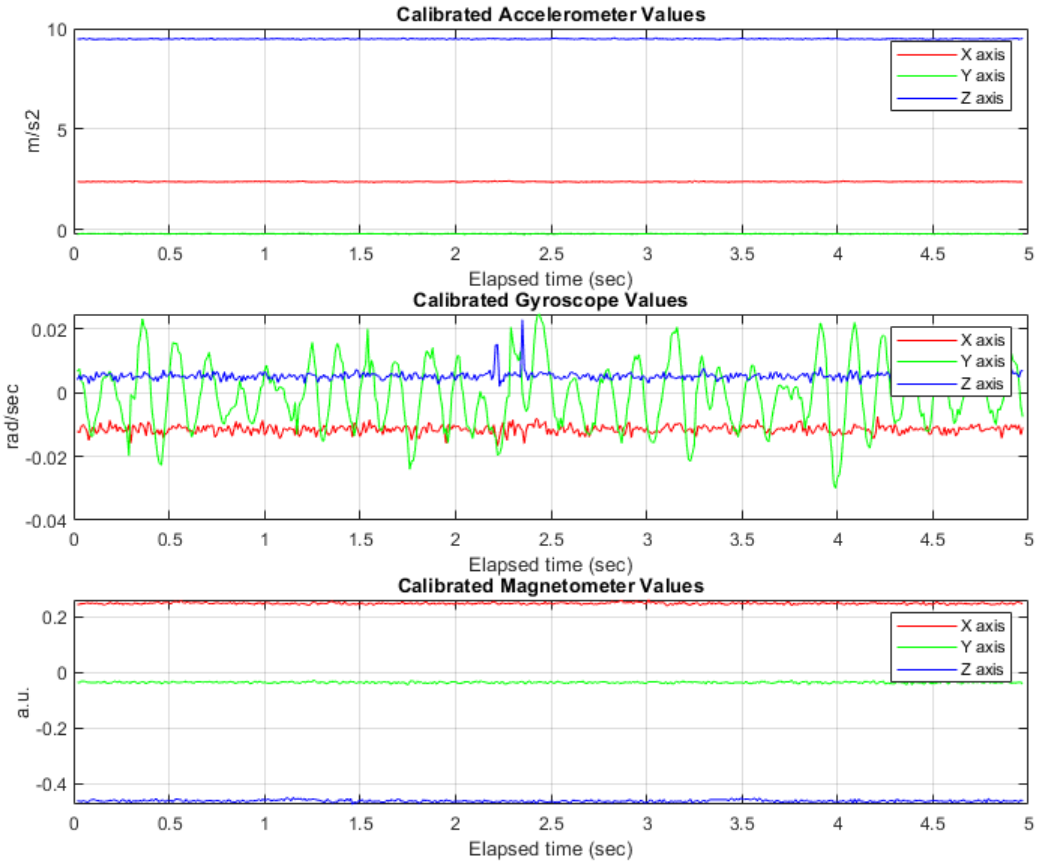


Figure 3.8 Calibrated accelerometer, gyroscope and magnetometer values obtained from MTi-3-DK

3.5 Robot Control Script

As mentioned in Chapter 2, the designed master program controls the UR10 robot through Script Level programming. After the movement record process is finished, resulted pose data is transformed to a script code and sent to URControl which URControl can execute.

Five functions of the URScript are used to control the robot: *movel*, *servoj*, *get_inverse_kin*, *set_tool_voltage* and *set_tool_digital_out*.

The *set_tool_voltage* function in URScript is used to set the voltage value for the tool (end effector). This function allows you to specify the voltage to which the tool should be set. The syntax typically involves providing the desired voltage value as an argument to the function.

The *set_tool_digital_out* function in URScript is used to control the state (ON or OFF) of a specific digital output on the robot's tool. Digital outputs on the tool are often used to interface with external devices, such as grippers, sensors, or other peripherals. In our system, digital output 0 is used to start and stop the spraying gun attached to robot's tool. The syntax for *set_tool_digital_out* typically involves specifying the output number and the desired state.

The *movel* function in URScript is used for linear motion. It commands the robot to perform a linear move to a specified pose in the Cartesian space. The *movel* command allows you to define both the position and orientation of the desired pose.

The *get_inverse_kin* function in URScript is used to calculate the inverse kinematics of the robot. It takes the target pose of the end-effector as input and returns the joint angles required to achieve that pose. The function is typically used to find a set of joint angles that positions the robot's end-effector at a desired location and orientation in its workspace.

The *servoj* function in URScript is used to control the robot's joints by specifying the desired joint positions. It is commonly used for tasks that require continuous, controlled motion of the robot's joints. The function takes the joint positions and additional parameters as inputs.

In the Figure 3.9, it is shown that how these functions are used during the programming the robot.

```
%% URScript functions
set_tool_voltage(12) ;
set_tool_digital_out(0,True));
movel(p[-0.588,0.574,0.593,-0.01254,-0.00420,0.019826],a=1.2, v=0.5);
servoj(get_inverse_kin(p[-0.588,0.574,0.593,-0.0132,-0.0043,0.0180]),t=0.040);
```

Figure 3.9 Examples of the URScript functions used to control the robot

In the Figure 3.9, the *set_tool_voltage* function is used to set the tool voltage to 12

volts, where *set_tool_digital_out* function is used to set digital output 0 on the tool to the "ON" state (true). With the *movel* function, the robot is commanded to move linearly to the specified pose with a custom acceleration (a) and velocity (v). And lastly, the *get_inverse_kin* function is used to calculate the joint angles required to reach the given target pose, and the *servoj* function is then used to smoothly move the robot to the calculated joint angles.

CHAPTER FOUR
EXPERIMENTAL RESULTS ON POSE ESTIMATON

This chapter presents the testing process of the movement record system, the obtained accuracy results, as well as the accuracy results of the sensor fusion algorithm, along with a comparison with its internal EKF and with the measured yaw angle by string-encoders.

4.1 Environment Created to Test Accuracy of the Movement Record System

After completing the design, manufacturing, and programming phases, the next step was to assess the system's accuracy. To achieve this, a ground-truth reference object was required to measure the pose of the teaching tool along with the designed movement record system. Robots are highly precise devices, allowing for the retrieval of the TCP's pose at any given time. Consequently, the robot used in the application cabinet was transported to the movement record cabinet, and with the help of the designed metal parts, the teaching tool was mounted to the TCP of the robot. This new test environment is shown in Figure 4.1.

Two different experiments have been conducted to test the accuracy of the movement record system. The first test aims to assess x, y, z, roll, pitch, and yaw measurement results independently by moving the robot's TCP in only one dimension at a time. This approach allows us to verify whether the calculations made by the master program are correct in each dimension. The second test involves to replay a pre-recorded operator's spraying movements by robot, and compare obtained movement record system results with the robot's actual TCP values.

Figure 4.1 Test environment

4.2 Accuracy Test of the Movement Record System

4.2.1 Testing Sensor System Measurement Accuracy in Each Dimension Independently

For illustrative purposes, Figure 4.2 presents a script code sent to URControl for executing robot movement solely in the yaw angle. The code begins by moving the robot to an initialization position to calibrate string encoders. Subsequently, the robot is directed to a position where the yaw angle is set to 0°. The script then commands the robot to loop three times between yaw angles -15° and +15° before returning to the 0° yaw angle position. Throughout these motions, the master program captures sensor data from both string encoders and IMU, calculating the TCP's pose, while simultaneously reading the actual positions from the robot.

Measurement results of sensor system in compare with the robot's actual TCP readings are shown in Figure 4.3, where error graphs, the difference between robot's TCP readings and the sensor system results, are shown in Figure 4.4. Table 4.1 shows

```
def onlyyaw():
set_tcp(p[-0.0235,0.0,0.2784,0.0,0.0,0.0]) $ initialization pose.
set_payload(0.5)
set_gravity([0.0, 0.0, 9.82])
$ 1 "Robot Program"
$ 2 "MoveL"
$ 3 "Waypoint_1"
movel([-1.2798, -1.8475, 2.5841, -2.3099, 1.5536, 2.7682], a=1.2, v=0.25) $ yaw 0
$ 4 "Wait: 1.0"
sleep(1.0)
$ 5 "Loop 3 times"
Loop_1 = 0
while (Loop_1 < 3):
$ 6 "Waypoint_2"
movel([-1.2724, -1.8572, 2.5900, -2.30615, 1.5536, 2.4990], a=1.2, v=0.25) $ yaw -15
$ 7 "Wait: 1.0"
sleep(1.0)
$ 8 "Waypoint_3"
movel([-1.2831, -1.8351, 2.5764, -2.3146, 1.5535, 3.0332], a=1.2, v=0.25) $ yaw +15
$ 9 "Wait: 1.0"
sleep(1.0)
Loop_1 = Loop_1 + 1
end
$ 10 "Waypoint_4"
movel([-1.2798, -1.8475, 2.5841, -2.3099, 1.5536, 2.7682], a=1.2, v=0.25) $ yaw 0
end
```

Figure 4.2 Script code that sent to URControl to test yaw angle independently

the RMS errors of the pose measurements.

Table 4.1 RMS errors of the pose measurements on independent movement test

Observed Data	**RMS Errors**
X [mm]	1.4054
Y [mm]	0.5842
Z [mm]	1.0868
IMU Roll [°]	0.5177
IMU Pitch [°]	0.1401
IMU Yaw [°]	0.2259
Meas. Yaw [°]	0.1331

As seen in the Table 4.1, sensor system accuracy on independent movement test is highly reliable.

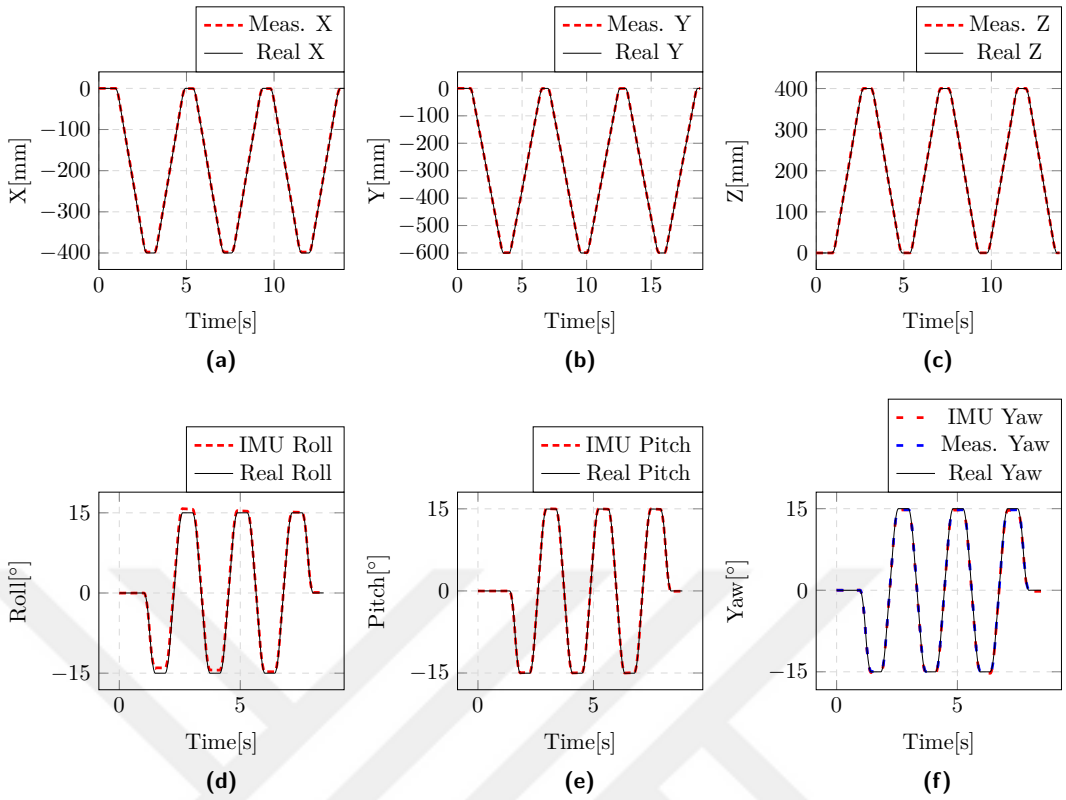


Figure 4.3 Pose measurements on independent movement test (a) X-axis position, (b) Y-axis position, (c) Z-axis position, (d) Roll angle, (e) Pitch angle, and (f) Yaw angle

4.2.2 Testing Sensor System Measurement Accuracy with Compound Movement Test

In the second experiment, motion data previously recorded while the operator was applying the spraying process is transmitted to the robot. This data was recorded before the robot was transported to the movement recording cabinet and includes approximately 35 seconds of motion with compound movements in all dimensions. To gain more information about the accuracy of the sensor system, the same motion data is replayed at different speeds by adjusting script commands to the robot. A speed of 100% represents the real speed of the operator.

Measurement results of sensor system in compare with the robot's actual TCP readings are shown in Figure 4.5, where error graphs are shown in Figure 4.6. Table 4.2 shows the RMS errors of the pose measurements in five different robot speed. 3D

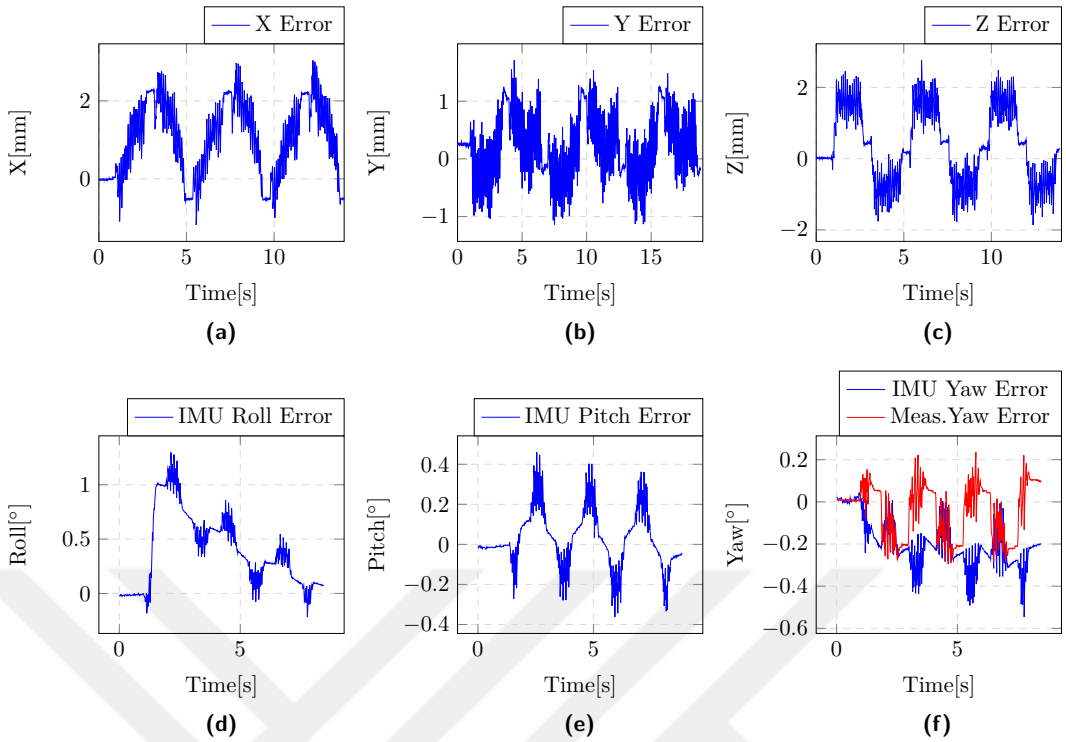


Figure 4.4 Pose errors on independent movement test (a) X-axis position error, (b) Y-axis position error, (c) Z-axis position error, (d) Roll angle error, (e) Pitch angle error, and (f) Yaw angle error

graph that shows the position data of the operator's spraying movement is shown in Figure 4.7.

Table 4.2 RMS error calculations of the second experiment

Observed Data	**RMS Errors**				
	20%	**40%**	**60%**	**80%**	**100%**
X [mm]	4.3398	4.2952	4.3013	4.2866	4.3603
Y [mm]	2.7378	2.7135	2.6621	2.7068	2.7323
Z [mm]	4.5901	4.6159	4.5780	4.5721	4.7349
IMU Roll [°]	0.5484	0.3009	0.2795	0.2657	0.3144
IMU Pitch [°]	0.4362	0.3606	0.4358	0.5180	0.5627
IMU Yaw [°]	7.4704	5.3151	3.7204	2.7342	2.0912
Meas. Yaw [°]	0.8195	0.7958	0.8306	0.8317	0.8390

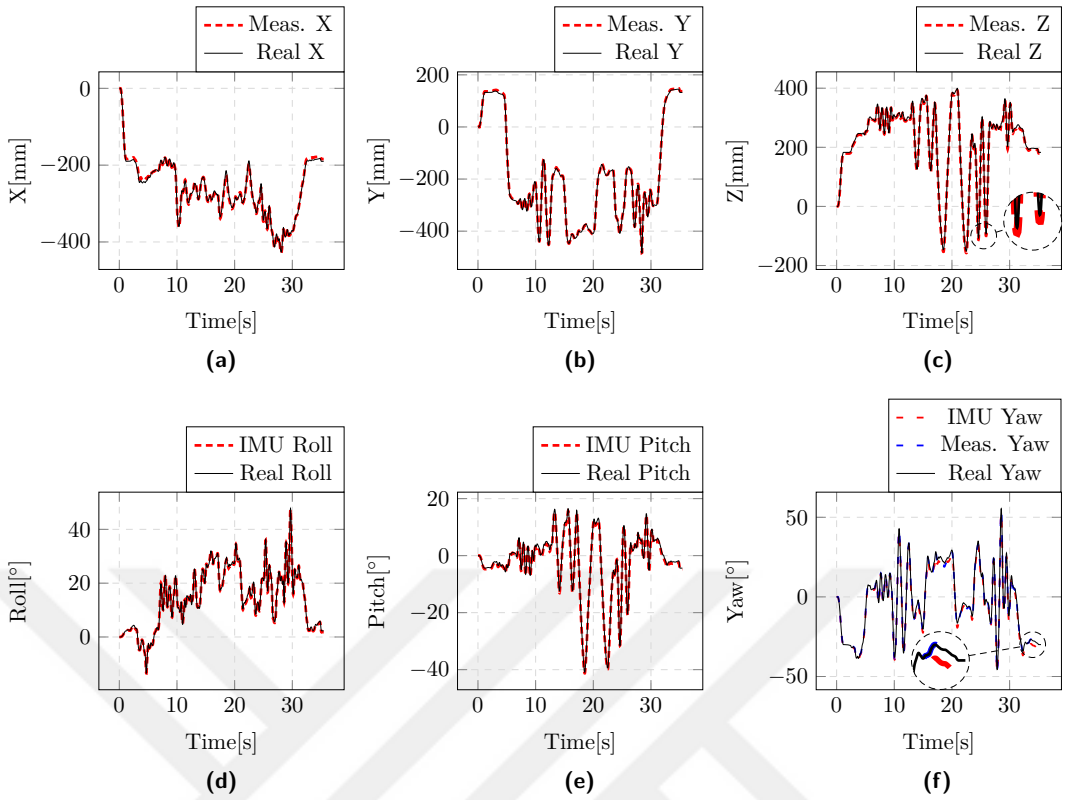


Figure 4.5 Pose measurements on the second experiment (a) X-axis position, (b) Y-axis position, (c) Z-axis position, (d) Roll angle, (e) Pitch angle, and (f) Yaw angle

4.2.3 Review of the results of the tests

When comparing the error results of position measurements between these two experiments, it is observed that the RMS errors in the second experiment are larger than those in the first experiment. RMS position estimation error in the first experiment is 1.87 mm, where it was 6.99 mm in the second experiment. The increase in errors may be attributed to differences in tensions on the different string lengths during continuous movement. Nevertheless, both test results showed that the position estimation accuracy of the sensor system is sufficient for the target application.

Similarly, the orientation estimation errors are larger in the second experiment, due to the higher orientation changes applied to the sensors in the IMU during complex movements. RMS IMU EKF orientation estimation error in the first experiment is 0.58° , where it was 2.19° in the second experiment, at 100% speed. The significant

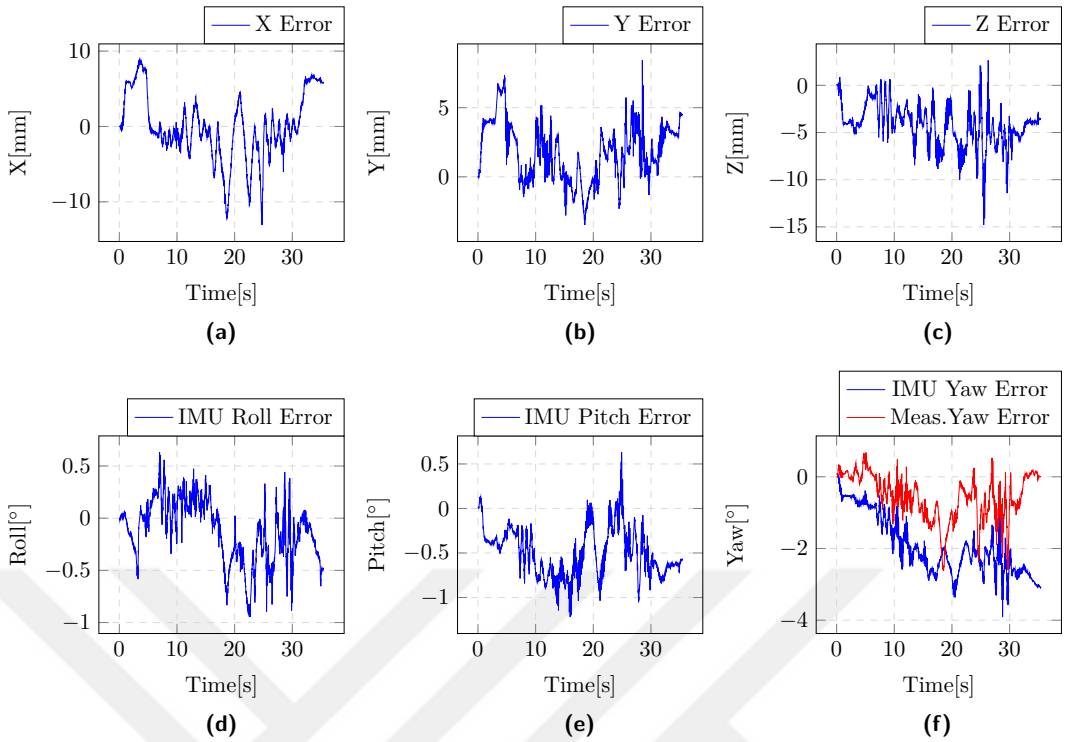


Figure 4.6 Pose errors on the second experiment (a) X-axis position error, (b) Y-axis position error, (c) Z-axis position error, (d) Roll angle error, (e) Pitch angle error, and (f) Yaw angle error

increase in the error occurred in the yaw angle output of the IMU, which is an expected outcome based on the limitations of the IMU, as described in Chapter 1. In addition to this, the RMS error on yaw angle estimation are dramatically increased when movement speed is reduced. For instance, the RMS error in yaw angle estimation during complex movement test at 100% speed was 2.09°, whereas it was 7.47° at 20% speed in the same motion. Roll and pitch angle estimations are slightly differ during all these tests, which are sufficient for the target application.

Despite the IMU EKF yaw angle estimation results are not always reliable according to the test results, the calculated yaw angle estimation, which is calculated by using string-encoders, results are almost constant in all the movement speed in second experiment. Calculated yaw angle estimation is 60% better than the IMU yaw angle estimation in 100% movement speed. If calculated yaw angle is used instead of IMU EKF yaw angle estimation, then RMS error of orientation estimation of the sensor system is reduced to 0.55 ° in the first experiment, and 1.06° in the second

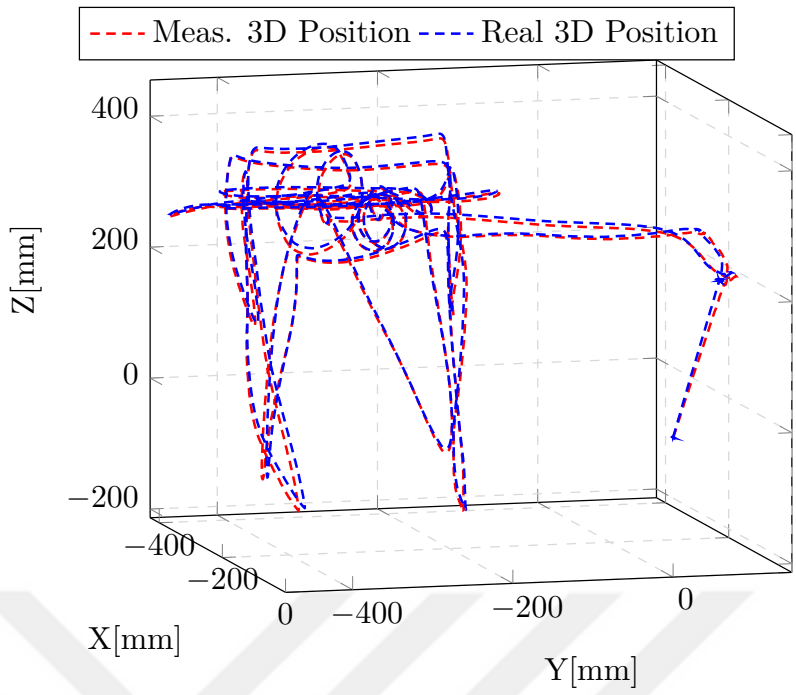


Figure 4.7 3D representation of the operator's movements during spraying application

experiment.

In contrast to the former results of this study (Yavuz et al., 2019), notable enhancements in both position and orientation accuracy are evident in the current version (Yavuz et al., 2021). These improvements stem from the implementation of an enhanced initialization procedure for the string encoders and the adoption of a new method for measuring yaw angles.

Referring to the findings presented in (Jiafan et al., 2009), the RMS errors of the movement record system are approximately 5 mm and 6°. Conversely, (Andrade-Cetto & Thomas, 2006) report larger errors, indicating RMS pose errors of around 200 mm and 9°, particularly in complex movement scenarios. In (Wang et al., 2017), a cable-driven parallel robot system is documented with RMS pose errors of 9.4 mm and 1.2°. However, pertinent details regarding movement characteristic, observation duration, and workplace size are not provided. Additionally, (Won et al., 2011) note a maximum position error of 37 mm, where there is no information about the orientation error. Upon comprehensive evaluation of these outcomes, it is evident

that the proposed system design yields superior accuracy and precision in both position and orientation estimations.

The system underwent real-time testing, which can be called as teleoperation. This test result revealing that the robot exhibits a time delay of 25 ms in mirroring the operator's movements. By enhancing the communication speed between URControl and the system program, it is feasible to achieve a reduction in this time delay. Specifically, for the UR10 robot, the time delay can potentially be minimized to approximately 8 ms. However, in the real world, a 25 ms delay is acceptable for many applications.

CHAPTER FIVE
EXPERIMENTAL WORKS AND RESULTS ON SENSOR FUSION ALGORITHM DESIGN

This chapter provides a comprehensive presentation of the experimental results obtained during the implementation and evaluation of the sensor fusion algorithm within the scope of this thesis.

In Chapter 3, the process of interpreting raw data from the ADIS16480 IMU is detailed. Subsequent to the conversion of raw data into physical units, this data becomes viable for utilization in sensor fusion algorithms.

The raw data employed in the sensor fusion algorithm is captured using the test environment outlined in Chapter 4. This approach enables a comparison between the results of the sensor fusion algorithm and the ground-truth readings obtained from the robot.

5.1 Studied Sensor Fusion Algorithm Types

Within the MATLAB SFTT, users have access to a diverse array of sensor fusion algorithms designed to facilitate a wide range of applications. In this study, the ImuFilter Function and AHRS function from the MATLAB SFTT are employed.

In addition to these built-in filters, our own designed Kalman Filter and Complementary Filters are utilized. The orientation estimation performance of these four algorithms have been tested using compound movement data, where this data described in Chapter 4.

5.1.1 ImuFilter Function of MATLAB SFTT

The ImuFilter function in the MATLAB Sensor Fusion and Tracking Toolbox is designed to create an IMU sensor fusion filter. This filter is utilized for sensor fusion applications involving IMU data, where readings from gyroscopes and accelerometers

are combined to determine pose of an object. In Figure 5.1, a MATLAB code script is presented, illustrating the creation of an ImuFilter object and its utilization with compound movement data as an input to the filter object.

```
%% ImuFilter Function of MATLAB SFTT

% assigning the parameters
SampleRate = 102.5;
dt = 1/SampleRate;
decim = 1;

fuse = imufilter('SampleRate',SampleRate,'DecimationFactor',decim,
'GyroscopeNoise',GyroNoise,'AccelerometerNoise',AccNoise);
%fuse = imufilter('SampleRate',SampleRate,'DecimationFactor',decim);

q = fuse(accelReadings,gyroReadings);
time = (0:decim:size(accelReadings,1)-1)/SampleRate;

% Transform quaternion to euler angles for seperate plot
[ImuFilter_eulerangles] = eulerd(q,'ZYX','frame');
ImuFilter_eulerangles = ImuFilter_eulerangles';
ImuFilter_Yaw = ImuFilter_eulerangles(1,:);
ImuFilter_Pitch = ImuFilter_eulerangles(2,:);
ImuFilter_Roll = ImuFilter_eulerangles(3,:);
```

Figure 5.1 MATLAB SFTT code segment for utilizing the ImuFilter function

5.1.2 AHRS Filter Function of MATLAB SFTT

The *ahrsfilter* function in the MATLAB Sensor Fusion and Tracking Toolbox is designed to implement an Attitude and Heading Reference System (AHRS) algorithm. AHRS is a sensor fusion technique that combines data from various sensors, such as accelerometers, gyroscopes, and magnetometers, to estimate the orientation (attitude) and heading of a device in three-dimensional space. In Figure 5.2, a MATLAB code script is presented, illustrating the creation of an ahrsfilter object and its utilization with compound movement data as an input to the filter object.

5.1.3 Kalman Filter Test

The mathematical background of Kalman filter has been explained in Section 2.3.8.1. The MATLAB code, which implements the mathematical equations of the Kalman Filter including prediction and update steps, is illustrated in Figure 5.3. The

```
%% AHRSfilter Function of MATLAB SFTT

fuse = ahrsfilter('SampleRate',SampleRate,'DecimationFactor',decim,'GyroscopeNoise',
GyroNoise,'AccelerometerNoise',AccNoise);
q = fuse(accelReadings,gyroReadings,magnReadings);
time = (0:decim:size(accelReadings,1)-1)/SampleRate;


% Transform quaternion to euler angles for seperate plot
[AHRS_eulerangles] = eulerd(q,'ZYX','frame');
AHRS_eulerangles = AHRS_eulerangles';
AHRS_Yaw = AHRS_eulerangles(1,:);
AHRS_Pitch = AHRS_eulerangles(2,:);
AHRS_Roll = AHRS_eulerangles(3,:);
```

Figure 5.2 MATLAB SFTT code segment for utilizing the ahrsfilter function

filter processes compound movement data as input, generating roll and pitch angles as outputs.

```
% Prediction step
prediction = A * prediction + B * [phi_dot, theta_dot]';
P = A * P * A' + Q;

% Update step
readings = [phi_hat_acc(i) theta_hat_acc(i)]';
y_tilde = readings - H * prediction;
S = R + H * P * H';
K = P * H' * (S^-1);
prediction = prediction + K * y_tilde;
P = (eye(4) - K * H) * P;
```

Figure 5.3 MATLAB code segment for utilizing the Kalman Filter

In the prediction step, A is the state transition matrix, representing the relationship between the current state and the predicted state in the absence of external inputs. B is the control-input matrix, accounting for the effect of control inputs on the state transition. P is the error covariance matrix of the state estimate, reflecting the uncertainty associated with the predicted state. Q is the process noise covariance matrix, representing the uncertainty or error introduced by the process model during prediction. *phi_dot* represents the rate of change (angular velocity) of the roll angle (*phi*), where *theta_dot* represents the rate of change (angular velocity) of the pitch angle (*theta*).

In the update step of the filter, *readings* is the actual measurement vector, typically

obtained from sensors, representing the observed roll *phi_hat_acc* and pitch *theta_hat_acc* angles. *y_tilde* is the innovation or measurement residual, which is the difference between the actual measurement and the predicted measurement based on the Kalman Filter state estimate. S is the innovation covariance, reflecting the uncertainty associated with the measurement. R is the measurement noise covariance matrix, representing the uncertainty or error in the measurement process. H is the measurement matrix that relates the state to the measurement. K is the Kalman Gain, determining the weight given to the difference between the actual measurement and the predicted measurement. It is calculated to minimize the estimation error.

5.1.4 Complementary Filter Test

The theoretical foundations of the Complementary filter are discussed in Section 2.3.8.3. In Figure 5.4, a MATLAB code script is presented, illustrating the implementation of the Complementary filter using compound movement data as an input. Consequently, the filter outputs roll and pitch angles.

```
%% Complementary Filter

alpha = 0.01;

phi_hat_complementary(i)   = (1 - alpha) * phi_hat_gyr_comp
+ alpha * phi_hat_acc(i);
theta_hat_complementary(i) = (1 - alpha) * theta_hat_gyr_comp
+ alpha * theta_hat_acc(i);
```

Figure 5.4 MATLAB code segment for utilizing the Complementary Filter

alpha is a blending factor or weight that determines the contribution of each sensor type to the final estimate. In this case, the gyroscope contributes with a weight of (1 - alpha), and the accelerometer contributes with a weight of alpha. *phi_hat_complementary* is the estimated roll angle obtained by blending the gyroscope-based estimate *phi_hat_gyr_comp* and the accelerometer-based estimate *phi_hat_acc*.

theta_hat_complementary is the estimated pitch angle obtained by blending the

gyroscope-based estimate *theta_hat_gyr_comp* and the accelerometer-based estimate *theta_hat_acc*.

5.2 Comparison of the Results of the Sensor Fusion Filters

In the Figure 5.5, roll angle estimations of presented sensor fusion algorithms are compared with the real roll data captured from UR10 robot arm. The AHRS filter function exhibits the best performance among these filters in terms of roll angle estimation, although the overall outcome of this filter still falls short of the desired accuracy.

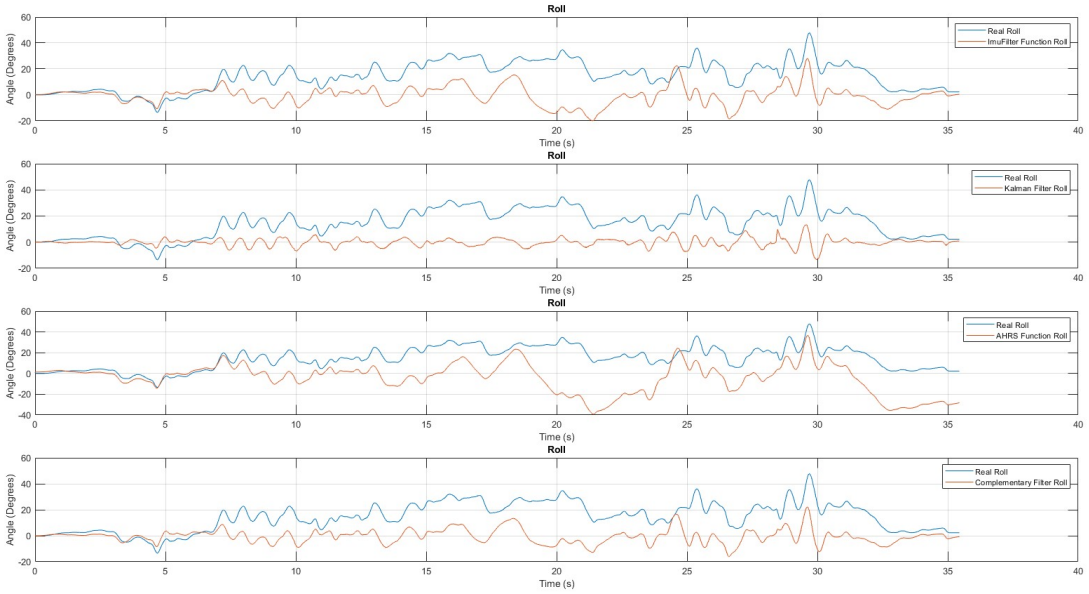


Figure 5.5 Comparison of roll angle estimations from the sensor fusion algorithms with real roll angle data

In the Figure 5.6, pitch angle estimations of presented sensor fusion algorithms are compared with the real pitch data captured from UR10 robot arm. The plots reveal that Imufilter and AHRS filter functions exhibit similar results, demonstrating lower errors compared to the roll angle estimations of these filters.

In the Figure 5.7, yaw angle estimation of AHRS filter is shown and compared with the real yaw data and calculated yaw angle data by using string-encoders. The plots

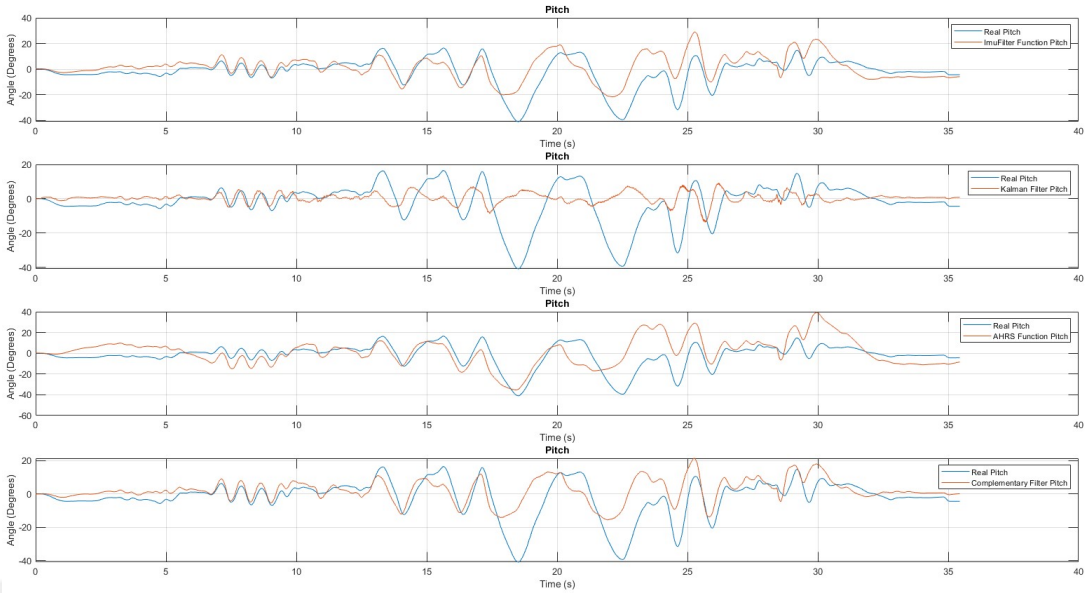


Figure 5.6 Comparison of pitch angle estimations from the sensor fusion algorithms with real pitch angle data

reveal that AHRS filter performed a relatively good performance in the first half of the movement, but then a bias error is formed.

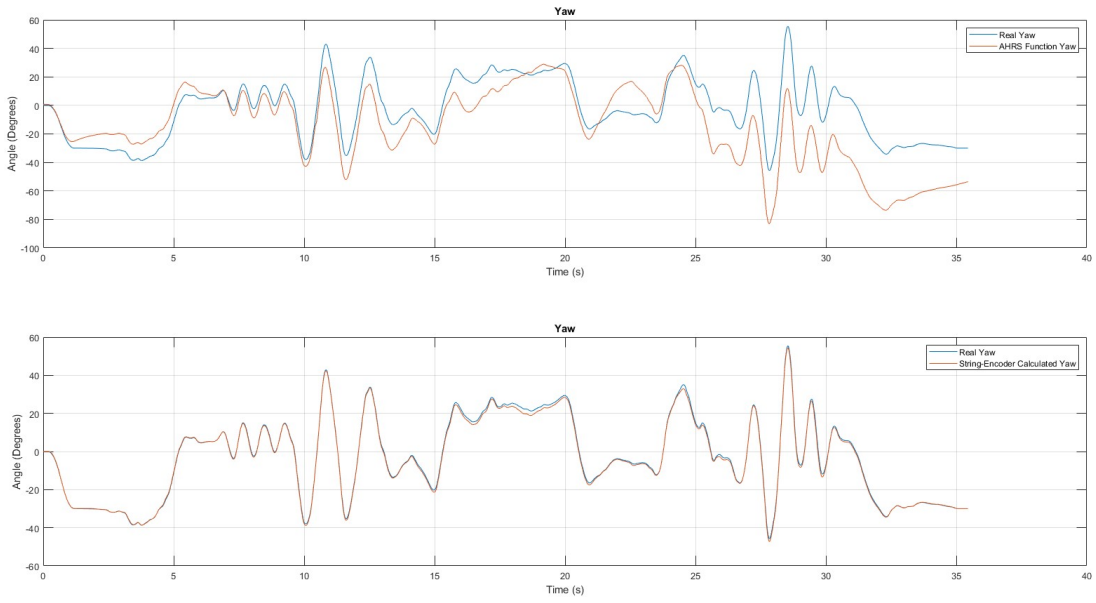


Figure 5.7 Comparison of yaw angle estimations from AHRS function and calculated yaw angle, with real yaw angle data

The examination of these figures reveals that the current sensor fusion algorithms

exhibit substantial estimation errors, indicating a need for improvement. In response to this, ongoing efforts are directed towards enhancing the algorithms. Additionally, there is an exploration into the integration of alternative fusion algorithms, including the Madgwick filter, along with the incorporation of existing and readily available MATLAB functions. This approach is part of a broader strategy aimed at refining the accuracy and performance of the sensor fusion process.

CHAPTER SIX
CONCLUSION

In this thesis, the aim is to address a challenge in the PP spraying task within the textile industry by employing a robot arm instead of a human for handling spraying tasks. To train the robot under the guidance of an expert operator, it is imperative to record the movements of the spraying gun during the spraying process. Therefore, a custom-designed pose measurement system is introduced. The purpose of this system is to estimate the pose of the teaching tool, serving as a replacement for the spraying gun.

The designed system comprises an IMU, six string-encoder position sensors, a UR10 robot manipulator, a spraying system, two custom controller boards, and a master system program.

The position estimation part is handled by using four string-encoder position sensors, along with calculation techniques such as the law of cosines and multilateration.

Orientation estimation is approached in three different ways. In the first method, Euler angle values are directly read from the IMU. In the second method, the yaw angle is measured using string-encoders. The third method involves calculating the Euler angles using different types of sensor fusion algorithms.

With the novel design, the robot is capable of replicating the operator's movement profiles, and the operator does not require any robot programming knowledge for this operation. This replication process can be conducted in real-time or offline.

Results from the conducted experimental tests have confirmed that the precision and accuracy of the designed system are suitable for tasks such as spraying, as well as for bleaching, painting, and palletizing. Moreover, the designed system exhibits superior accuracy and precision when compared to analogous systems employing the same types of sensors.

Furthermore, the experimental tests reveal that the performance of the designed and tested sensor fusion algorithms needs improvement. Ongoing studies are focused on enhancing the code of the sensor fusion algorithm.

The future work of this thesis aims to enhance the performance of the designed sensor fusion algorithm and publish a paper covering the algorithm along with a comparison to existing sensor fusion algorithms.

REFERENCES

Abyarjoo, F., Barreto, A., Cofino, J., & Ortega, F. R. (2015). Implementing a sensor fusion algorithm for 3d orientation detection with inertial/magnetic sensors. In *Innovations and advances in computing, informatics, systems sciences, networking and engineering*, *Springer*, 305–310. https://doi.org/10.1007/978-3-319-06773-5_41.

Akgun, B., Cakmak, M., Yoo, J. W., & Thomaz, A. L. (2012). Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, *ACM*, 391–398. https://doi.org/10.1145/2157689.2157815.

Alatise, M., & Hancke, G. (2017). Pose estimation of a mobile robot based on fusion of imu data and vision data using an extended kalman filter. *Sensors*, *17*(10), 2164. https://doi.org/10.3390/s17102164.

Analog Devices Inc. (2017). *Datasheet of ADIS16480, Rev. F. (2018, August 22)*. http://www.analog.com/media/en/technical-documentation/data-sheets/ADIS16480.pdf.

Andrade-Cetto, J., & Thomas, F. (2006). Wire-based tracking using mutual information. In *Advances in Robot Kinematics*, 3–14.

Andrade-Cetto, J., & Thomas, F. (2008). A wire-based active tracker. *IEEE transactions on robotics*, *24*(3), 642–651. https://doi.org/10.1109/TRO.2008.924260.

Aoyagi, S., Kohama, A., Nakata, Y., Hayano, Y., & Suzuki, M. (2010). Improvement of robot accuracy by calibrating kinematic model using a laser tracking system-compensation of non-geometric errors using neural networks and selection of optimal measuring points using genetic algorithm. In *2010 IEEE/RSJ International conference on intelligent robots and systems*, *IEEE*, 5660–5665. https://doi.org/10.1109/IROS.2010.5652953.

Atac, R., & Foxlin, E. (2013). Scorpion hybrid optical-based inertial tracker (hobit). In *Head-and Helmet-Mounted Displays XVIII: Design and Applications*, vol. 8735, *International Society for Optics and Photonics*, 873502. https://doi.org/10.1117/12.2012194.

Barraza-Madrigal, J., Muñoz-Guerrero, R., Leija-Salas, L., & Ranta, R. (2014). Instantaneous position and orientation of the body segments as an arbitrary object in 3d space by merging gyroscope and accelerometer information. *Revista mexicana de ingeniería biomédica*, *35*(3), 241–252.

Božek, P., Al Akkad M, A., Blištan, P., & Ibrahim N, I. (2017). Navigation control and stability investigation of a mobile robot based on a hexacopter equipped with an integrated manipulator. *International Journal of Advanced Robotic Systems*, *14*(6), 1729881417738103. https://doi.org/10.1177/1729881417738103.

Chow, J. C. (2018). Statistical sensor fusion of a 9-dof mems imu for indoor navigation. *arXiv preprint arXiv:1802.04388*. https://doi.org/10.48550/arXiv.1802.04388.

Gao, M., Ye, L., & Yan, Y. (2015). Design and implementation of teach pendant for six degrees of freedom industrial robot. In *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, *IEEE*, 1929–1933. https://doi.org/10.1109/IMCCC.2015.409.

Georgy, J., Karamat, T., Iqbal, U., & Noureldin, A. (2011). Enhanced mems-imu/odometer/gps integration using mixture particle filter. *GPS solutions*, *15*(3), 239–252. https://doi.org/10.1007/s10291-010-0186-4.

Guo, S., Wu, J., Wang, Z., Qian, J., et al. (2017). Novel marg-sensor orientation estimation algorithm using fast kalman filter. *Journal of Sensors*, *2017*. https://doi.org/10.1155/2017/8542153.

Jiafan, Z., Jinsong, L., Liwei, Q., & Dandan, Z. (2009). Kinematic analysis of a 6-dof wire-based tracking device and control strategy for its application in robot easy programming. In *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, *IEEE*, 1591–1596. https://doi.org/10.1109/ROBIO.2009.5420394.

Kilin, A., Bozek, P., Karavaev, Y., Klekovkin, A., & Shestakov, V. (2017). Experimental investigations of a highly maneuverable mobile omniwheel robot. *International Journal of Advanced Robotic Systems*, *14*(6), 1729881417744570. https://doi.org/10.1177/1729881417744570.

Kirsch, S. R., Schilling, C., & Brunner, G. (2006). Assesment of metallic distortions of an electromagnetic tracking system. In *Medical Imaging 2006: Visualization, Image-Guided Procedures, and Display*, vol. 6141, *International Society for Optics and Photonics*, 61410J. https://doi.org/10.1117/12.654768.

Madgwick, S. O., Harrison, A. J., & Vaidyanathan, R. (2011). Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE international conference on rehabilitation robotics*, *IEEE*, 1–7. https://doi.org/10.1109/ICORR.2011.5975346.

Meina, M., Krasuski, A., & Rykaczewski, K. (2015). Model fusion for inertial-based personal dead reckoning systems. In *2015 IEEE Sensors Applications Symposium (SAS)*, *IEEE*, 1–6. https://doi.org/10.1109/SAS.2015.7133658.

Merlet, J.-P. (2008). Kinematics of the wire-driven parallel robot marionet using linear actuators. In *2008 IEEE International Conference on Robotics and Automation*, *IEEE*, 3857–3862. https://doi.org/10.1109/ROBOT.2008.4543803.

Moeslund, T. B., & Granum, E. (2001). A survey of computer vision-based human motion capture. *Computer vision and image understanding*, *81*(3), 231–268. https://doi.org/10.1006/cviu.2000.0897.

Nützi, G., Weiss, S., Scaramuzza, D., & Siegwart, R. (2011). Fusion of imu and vision for absolute scale estimation in monocular slam. *Journal of intelligent & robotic systems*, *61*(1-4), 287–299. https://doi.org/10.1007/s10846-010-9490-z.

Pirník, R., Hruboš, M., Nemec, D., Mravec, T., & Božek, P. (2015). Integration of inertial sensor data into control of the mobile platform. In *Federated Conference on Software Development and Object Technologies*, *Springer*, 271–282. https://doi.org/10.1007/978-3-319-46535-7_21.

Sick AG (2017). *Online datasheet of EcoLine PFG13-A1CM0544 Wire Draw Encoders. (2019, March 7)*. https://cdn.sick.com/media/pdf/1/21/421/dataSheet_PFG13-A1CM0544_1061015_en.pdf.

Thomas, F., Ottaviano, E., Ros, L., & Ceccarelli, M. (2002). Uncertainty model and singularities of 3-2-1 wire-based tracking systems. In *Advances in Robot Kinematics*, 107–116.

Thomas, F., Ottaviano, E., Ros, L., & Ceccarelli, M. (2005). Performance analysis of a 3-2-1 pose estimation device. *IEEE Transactions on Robotics*, *21*(3), 288–297. https://doi.org/10.1109/TRO.2004.840894.

Tomaszewski, D., Rapinski, J., & Smieja, M. (2015). Analysis of the noise parameters and attitude alignment accuracy of ins conducted with the use of mems-based integrated navigation system. *Acta Geodynamica et Geromaterialia*, *12*(2), 197–209. https://doi.org/10.13168/AGG.2015.0017.

Universal Robots (2016). *The URScript Programming Language, Version 3.8. (2019, March 7)*. https://s3-eu-west-1.amazonaws.com/ur-support-site/45989/scriptManual.pdf.

Wang, H., Gao, T., Kinugawa, J., & Kosuge, K. (2017). Finding measurement configurations for accurate robot calibration: Validation with a cable-driven robot. *IEEE Transactions on Robotics*, *33*(5), 1156–1169. https://doi.org/10.1109/TRO.2017.2707562.

Won, S.-h., Melek, W., & Golnaraghi, F. (2008a). Position and orientation estimation using kalman filtering and particle diltering with one imu and one position sensor. In *2008 34th Annual Conference of IEEE Industrial Electronics*, *IEEE*, 3006–3010. https://doi.org/10.1109/IECON.2008.4758439.

Won, S.-h., Parnian, N., Golnaraghi, F., & Melek, W. (2008b). A quaternion-based tilt angle correction method for a hand-held device using an inertial measurement unit. In *2008 34th Annual Conference of IEEE Industrial Electronics*, *IEEE*, 2971–2975. https://doi.org/10.1109/IECON.2008.4758433.

Won, S.-H. P., Golnaraghi, F., & Melek, W. W. (2009a). A fastening tool tracking system using an imu and a position sensor with kalman filters and a fuzzy expert system. *IEEE Transactions on Industrial Electronics*, *56*(5), 1782–1792. https://doi.org/10.1109/TIE.2008.2010166.

Won, S.-h. P., Melek, W., & Golnaraghi, F. (2009b). A fastened bolt tracking system for a hand-held tool using an inertial measurement unit and a triaxial magnetometer. In *2009 35th Annual Conference of IEEE Industrial Electronics*, *IEEE*, 2703–2708. https://doi.org/10.1109/IECON.2009.5415430.

Won, S.-h. P., Melek, W., & Golnaraghi, F. (2011). Fastening tool tracking system using a kalman filter and particle filter combination. *Measurement Science and Technology*, *22*(12), 125108. https://doi.org/10.1088/0957-0233/22/12/125108.

Won, S.-h. P., Melek, W. W., & Golnaraghi, F. (2009c). A kalman/particle filter-based position and orientation estimation method using a position sensor/inertial measurement unit hybrid system. *IEEE Transactions on Industrial Electronics*, *57*(5), 1787–1798. https://doi.org/10.1109/TIE.2009.2032431.

Xsens Technologies B.V. (2016). *Datasheet of MTi-1 series, Rev. D. (2016, December 5)*. https://www.xsens.com/hubfs/Downloads/Manuals/MTi-1-series-datasheet.pdff.

Yavuz, E., Senol, Y., Ozcelik, M., & Aydin, H. (2019). Design of a string-encoder and imu based 6d pose measurement system. *2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 1–6. https://doi.org/10.1109/ECAI46879.2019.9042021.

Yavuz, E., Şenol, Y., Özçelik, M., & Aydın, H. (2021). Design of a string encoder-and-imu-based 6d pose measurement system for a teaching tool and its application in teleoperation of a robot manipulator. *Journal of Sensors*, *2021*, 1–17. https://doi.org/10.1155/2021/6678673.

Zhang, S., Wang, W., & Jiang, T. (2020). Wifi-inertial indoor pose estimation for

micro aerial vehicles. *IEEE Transactions on Industrial Electronics*. https://doi.org/10.1109/TIE.2020.2984457.

Zhao, H., & Wang, Z. (2012). Motion measurement using inertial sensors, ultrasonic sensors, and magnetometers with extended kalman filter for data fusion. *IEEE Sensors Journal*, *12*(5), 943–953. https://doi.org/10.1109/JSEN.2011.2166066.