INVESTIGATION OF ADAPTIVE KALMAN FILTER TECHNIQUES FOR SENSOR FUSION APPLICATIONS

SENSÖR FÜZYON UYGULAMALARI İÇİN ADAPTİF KALMAN FİLTRESİ TEKNİKLERİNİN İNCELENMESİ

MUSTAFA DOĞRU

ASSOC. PROF. DR. İSMAIL UYANIK

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Master of Science

in Electrical and Electronics Engineering

September 2023

ABSTRACT

INVESTIGATION OF ADAPTIVE KALMAN FILTER TECHNIQUES FOR SENSOR FUSION APPLICATIONS

Mustafa DOĞRU

Master of Science , Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. İsmail UYANIK

September 2023, 80 pages

Sensor fusion is a concept encountered in many different disciplines today. In its simplest definition, this concept aims to combine information from different sources using appropriate methods. In this thesis, a simulation environment and experimental setup utilizing LiDAR and a camera have been established. A closed-loop model was employed in the experimental setups to test multi-sensor algorithms. Initially, a superposition-based tracking behavior was examined using the closed-loop model. Using this method, sensor data was fused by weighting it with a constant value. Subsequently, a Kalman filter was utilized to enhance state estimation under different conditions. The error covariance provided by the Kalman filter was used for weighting. Independent of the general noise conditions of the Kalman filter, the system's performance was examined using different measurement noise estimation algorithms found in the literature. Thus, the error resulting from the position estimation of the sensors could be dynamically determined. By dynamically adjusting the weights, an effective result was achieved in cases where environmental noise varied, and as a consequence, the system's tracking performance was improved.

Keywords: Kalman Filter, Sensor Fusion, Robotics, State Estimation, Tracking,

ÖZET

SENSÖR FÜZYON UYGULAMALARI İÇİN ADAPTİF KALMAN FİLTRESİ TEKNİKLERİNİN İNCELENMESİ

Mustafa DOĞRU

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği

Danışman: Assoc. Prof. Dr. İsmail UYANIK

Haziran 2023, 80 sayfa

Sensör füzyon günümüzde birçok farklı disiplinlerde karşımıza çıkan bir konseptir. Bu konsept en yalın tanımıyla farklı kaynaklardan gelen bilgileri uygun yöntemlerle birleştirmeyi amaçlamaktadır. Bu tezde LiDAR ve kameranın kullanıldığı bir simülasyon ortamı ve deney düzeneği geliştirilmiştir. Bu ortamlarda sensör füzyon algoritmalarını deneyebilmek için bir kapalı döngü modeli kullanılmıştır. Kapalı döngü modeli ile çalışan deney düzeneği üzerinden önce süper pozisyon tabanlı bir takip davranışı incelenmiştir. Bu yöntemde sensör verileri sabit bir değer ile ağırlıklandırılarak füzyon işlemi gerçekleştirilmiştir. Ardından sistemin farklı koşullar altında durum kestirimini iyileştirebilmek için Kalman filtresi kullanılmıştır. Kalman filtresi tarafından sağlanan hata kovaryansı ağırlıklandırma için kullanılmıştır. Literatürde yer alan farklı ölçüm gürültü kestirimi algoritmaları kullanılarak Kalman filtresinin performansı incelenmiştir. Bu sayede sensörlerin pozisyon kestirimi sonucunda oluşan hata dinamik olarak kestirilebilmektedir. Dinamik olarak değiştirilen ağırlıklar ortam gürültüsünün farklılaştığı durumlarda başarılı bir sonuç vererek sistemin takip performansının iyileşmesini sağlamıştır.

Keywords: Kalman Filtresi, Sensör Füzyon, Robotik, Durum Kestirimi, Takip

ACKNOWLEDGEMENTS

I want to begin by thanking my supervisor, Dr. Ismail UYANIK, for all of his advice, encouragement, and support during my studies. He was always supportive and helped me as I went through my graduate education.

I also want to express my gratitude to the esteemed members of my thesis jury, Dr. Yakup ÖZKAZANÇ, Dr. Mustafa Mert ANKARALI, Dr. Şölen Kumbay YILDIZ, and Dr. Barış YÜKSEKKAYA, for their invaluable feedback. I owe a debt of gratitude to every teacher I had at Hacettepe University while pursuing my undergraduate and graduate degrees.

Then, I want to continue by thanking Dr. Burcu ÜNLÜTABAK for her support and guidance during my thesis period.

Last but not least, I would want to express my sincere gratitude to my parents for their unwavering support, love, and inspiration throughout my entire life.

The work presented in this thesis was supported by the Scientific and Technological Research Council of Türkiye (TÜBİTAK) through project 122E464 awarded to Assoc. Prof. İsmail UYANIK.

CONTENTS

Page

TABLES

Page

FIGURES

ABBREVIATIONS

LiDAR	:	**L**aser **I**maging **D**etection **A**nd **R**anging
AI	:	**A**rtificial **I**ntelligence
PDF	:	**P**robability **D**ensity **F**unction
KF	:	**K**alman **F**ilter
MCC-KF	:	**M**aximum **C**orrentropy **C**riterion **K**alman **F**ilter
ROS	:	**R**obot **O**perating **S**ystem
URDF	:	**U**niversal **R**obot **D**escription **F**ormat
MATLAB	:	**MAT**rix **LAB**oratory
PID	:	**P**roportional **I**ntegral **D**erivative
UART	:	**U**niversal **A**synchronous **R**eceiver **T**ransmitter
FPS	:	**F**rame **P**er **S**econd
RMSE	:	**R**oot **M**ean **S**quare **E**rror
FFT	:	**F**ast **F**ourier **T**ransform

1. INTRODUCTION

This thesis acts as a foundation for the implementation of biological integration algorithms in engineering systems. For this purpose, we propose a robotic system that is the equivalent to weakly electric fish' refuge tracking behavior [1–3]. To briefly explain the biological experiment setup, the electric fish are placed in a refuge and the refuge is moved when the environmental conditions are different (dark, light, etc.). It performs a tracking behavior by processing the data obtained by the fish with electrosensory and visual information during movement. Based on this, a two-dimensional moving robot assembly including LiDAR and camera was prepared. In this setup, there are two different objects that each sensor can track separately. Studies have been carried out on this experimental setup. Using this experimental setup, biology-inspired multi-sensory integration methods were compared with sensor fusion techniques in the literature, these methods were applied to a robotic system, and their performances were compared.

1.1. Scope Of The Thesis

Sensor fusion is a process that involves combining data from multiple sensors to obtain a more accurate and comprehensive understanding of the environment or a particular phenomenon. It aims to overcome the limitations and uncertainties associated with individual sensors by leveraging the strengths of different sensor types and integrating their measurements [4–6].

When the sensory systems of living organisms are examined, it is observed that throughout the process of evolution, their bodies are equipped with different sensory structures (such as eyes, ears, nose, etc.) that can perceive multi-modal signals (light, sound, smell, etc.) present in their surroundings. Multiple sensory structures can be stimulated by the same source. In this case, the central nervous system (CNS) of organisms performs a series of integration processes to enhance the reliability of information and simplify complex signals [7, 8].

Signals in the integration stage can have different characteristic features, and the information reaching sensory organs may not be simultaneous. Additionally, these acquired signals can also encompass different types of distortions. In such cases, the CNS can combine this information in the most suitable way to extract a sensory representation of the environment. There are many biological behaviors through which this phenomenon can be studied. For example, when animals like bees want to feed on nectar, they control their behavior using the olfactory signals and visual information they receive from their environment [9], weakly electric fish use information from vision, electrosense, and mechanosense when capturing prey [10], while a moth is feeding on a flower, it constantly updates its position according to the movement of the flower [11] or while the cockroaches are following a wall, they try to regulate the error difference by perceiving the position of the wall and their own position with the help of their eyes and antennae [12].

When we examine today's engineering world, it is possible to come across sensor fusion applications in many different fields such as autonomous vehicles, internet of things (IoT), computer vision [13–15] because sensor fusion has many advantages. We can summarize these advantages as follows [16].

- Increased Accuracy and Reliability: By integrating measurements from different sources, sensor fusion can provide more accurate and reliable information about the environment or observed phenomenon.
- Improved Perception and Understanding: By combining data from different types of sensors, sensor fusion can provide a more comprehensive perception of the environment. Each sensor type has its strengths and weaknesses, and sensor fusion allows the system to leverage the best aspects of each sensor. This leads to a better understanding of the environment, objects, events, or phenomena being observed.
- Robustness to Sensor Failures: In real-world scenarios, sensors may encounter failures or limitations due to various factors such as occlusions, environmental conditions, or technical issues. Sensor fusion allows for redundancy by using multiple sensors, which

improves the system's robustness. If one sensor fails or provides unreliable data, other sensors can compensate and maintain the overall performance.

From an engineering perspective, to benefit from the advantages mentioned above, it is necessary to correctly implement a series of steps for the sensor fusion process. While these steps may vary across applications, some of these steps are common to many applications. As shown in Figure 1.1 we can briefly express these steps.

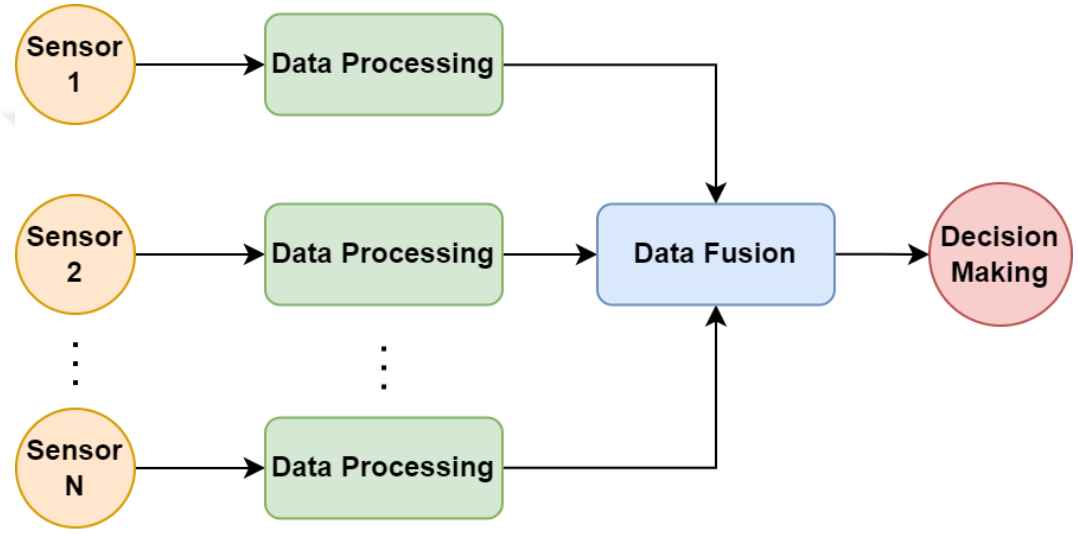

Figure 1.1 Simple Diagram of Sensor Fusion Process

The first step is to select the most suitable sensors for the application. These sensors can provide data in various forms. For example, in this thesis, LiDAR and a camera are used to estimate position. Here, while LiDAR provides distance information, the camera provides visual information. The obtained data is made available for sensor fusion in the data processing step. This may involve cleaning the data and removing noise. With the help of various algorithms, the data is fused and transferred to the decision-maker. The output generated by the decision-maker can be used to control a robot, navigate a vehicle, or identify an object. The main focus of this thesis will be on what kind of algorithms are used in the data fusion step.

The choice of sensor fusion algorithm depends on the specific application, the characteristics of the sensors, the available information, and the desired performance objectives. When the

literature is examined, it is seen that many algorithms are used. Some of these algorithms are given in Table 1.1 as examples [17, 18].

Estimation Methods	**AI Methods**	**Inference Methods**
Non-Recursive: • Weighted Average • Least Square Recursive: • Kalman Filter • Extended Kalman Filter	• Artificial Neural Network • Fuzzy Logic	• Dempster-Shafer Method • Bayesian Inference

Table 1.1 Sensor Fusion Methods

For the simplest fusion method, we can use the sensor data by weighting them at certain ratios. If we use this method, the errors in the sensor data will also be weighted. If we do not know the amount of corruption in the sensor data, this method will give very unsuccessful results. Instead, using methods such as the Kalman Filter, which aims to statistically combine the sensor data in the most optimal way, enables us to reach more reliable results by dynamically parameterizing the accuracy of the fusion method. We used these sensor fusion methods in our thesis based on this advantageous situation.

1.2. Contributions

- A simulation environment has been prepared where sensor fusion algorithms can be tested independently of hardware.
- We propose a novel experimental system in which LiDAR data and camera images can be combined and tested in a controlled, measurable manner during a target-tracking mission.

- Application of dynamic weighting methods inspired by biology to a physical system on an experimental setup.
- With the help of the experimental setup, it can be examined how the performance of the sensor fusion is affected by the independent interference factors of the LiDAR and camera data.

1.3. Organization

The organization of the thesis is as follows:

- Chapter 1 includes the general scope of the thesis, the explanation of the concepts within this scope, and the contributions of the thesis.
- Chapter 2 explains some concepts to understand the thesis better.
- Chapter 3 describes the simulation environment.
- Chapter 4 provides a detailed explanation of the experimental setup created for the thesis studies.
- Chapter 5 describes the system models required to test sensor fusion algorithms.
- Chapter 6 the data obtained from the test results are evaluated.
- Chapter 7 states the summary of the thesis and possible future directions.

2. BACKGROUND AND OVERVIEW

2.1. Bayesian State Estimation

Consider discrete time dynamic state space model with contain hidden state x_t and measurement state y_t. This system generally can be express as equation (1)

$$\begin{aligned} x_{t+1} &= f(x_t) + w_t \\ y_t &= h(x_t) + v_t \end{aligned} \tag{1}$$

$x_t \in \mathbb{R}^{d_x}$ is represent target state vector of order d_x at time t, $w_t \in \mathbb{R}^{d_x}$ is related noise with target state (or process noise). Process noise represent lack of knowledge about system dynamics. $y_t \in \mathbb{R}^{d_y}$ is represent measurement state vector of order d_y at time t, $v_t \in \mathbb{R}^{d_y}$ is related noise with measurement state (or measurement noise). Measurement noise give information about uncertainty of observed data [19].

Bayesian state estimation provides to estimate target state using measurements. Set of target states can be represented with $x_{1:t} = [x_1, x_2, x_3....x_t]$, set of measurement state can be represent with $y_{1:t} = [y_1, y_2, y_3....y_t]$. Measurements y observable of hidden Markov process and target states x_t assumed that unobservable. This situation shown as Figure 2.1

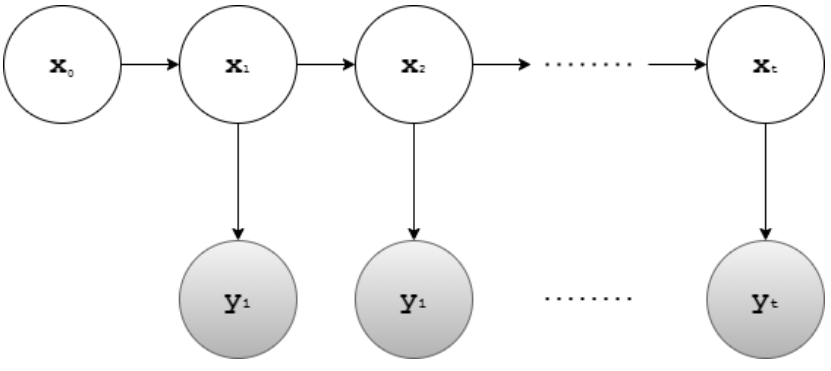

Figure 2.1 Hidden Markov Process of Bayesian State Estimation

Under Markov assumption probability of the current state only depend on previous state and can be written as (2)

$$p(x_t|x_{t-1}, x_{t-2}, x_{t-3}, ..., x_0) = p(x_t|x_{t-1}) \tag{2}$$

Similarly probability of the measurement state only depend on target state and can be written as equation (3)

$$p(y_t|x_t, x_{t-1}, x_{t-2}, ..., x_0) = p(y_t|x_t) \tag{3}$$

At this point problem can be express as a recursively estimate the posterior probability density function (pdf) of $p(x_t|y_t)$ starting from $p(x_0) = p(x_0|z_0)$. Recursive estimate process consists of prediction and update steps. Prediction-Update procedure firstly predict $p(x_t|y_{t-1})$ by using $p(x_{t-1}|y_{t-1})$. If we have new measurement y_t then we can update $p(x_t|y_{t-1})$ to $p(x_t|y_t)$. This sequence is illustrated as equation (4)

$$p(x_{t-1}|y_{1:t-1}) \xrightarrow{(Predict)} p(x_t|y_{1:t-1}) \xrightarrow{(Update)} p(x_t|y_{1:t}) \xrightarrow{(Predict)} p(x_{t+1}|y_{1:t}) \tag{4}$$

We can express predicted pdf as (5)

$$p(x_t|y_{1:t-1}) = \int p(x_t, x_{t-1}|y_{1:t-1})\, dx_{t-1} \tag{5}$$

From $P(A, B) = P(A|B)P(B)$ (5) equation can be written as (6)

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1}, y_{1:t-1})p(x_{t-1}|y_{1:t-1})\, dx_{t-1} \tag{6}$$

According to Markov process $p(x_t|x_{t-1}, y_{1:t-1}) = p(x_t|x_{t-1})$ equation (6) reduced as equation (7)

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1}p(x_{t-1}|y_{1:t-1})\, dx_{t-1} \tag{7}$$

Also If we get new measurement y_t, $p(x_t|y_{1:t-1})$ can be update to $p(x_t|y_{1:t})$ as equation (8)

$$\begin{aligned} p(x_t|y_{1:t}) &= \frac{p(x_t, y_t)}{p(y_t)} = \frac{p(x_t, y_t, y_{1:t-1})}{p(y_t, y_{1:t-1})} \\ &= \frac{p(y_t|x_t, y_{1:t-1})p(x_t, y_{1:t-1})}{p(y_t|y_{1:t-1})} \\ &= \frac{p(y_t|x_t)p(x_t, y_{1:t-1})}{p(y_t|y_{1:t-1})} \end{aligned} \tag{8}$$

Update equation and prediction equation have some computational difficulties like calculating likelihood function $p(y_t|x_t)$ or calculating $p(x_t|x_{t-1})$. These calculations can be made easily under certain assumptions.

2.2. Kalman Filter

Kalman filter assume that prediction pdf $p(x_t|y_{1:t-1})$ and update pdf $p(x_t|y_{1:t})$ are both Gaussian. In addition, It also assumed that state space model is linear. Given this assumptions, equation (1) discrete form can be rewritten as equation (9)

$$\begin{aligned} x_{k+1} &= Ax_k + w_k \\ y_k &= Cx_k + v_k \end{aligned} \tag{9}$$

$A \in \mathbb{R}^{d_x x d_x}$, $w_k \sim \mathcal{N}(w_k; 0, Q)$ process noise $Q \geq 0 \in \mathbb{R}^{d_x x d_x}$, $C \in \mathbb{R}^{d_y x d_x}$, $v_k \sim \mathcal{N}(v_k; 0, R)$ measurement noise $R \geq 0 \in \mathbb{R}^{d_y x d_y}$

After this point we can only use mean and covariance of x_k for predict and update process rather than calculating pdf. In this case, the equations in the previous section can be rewritten as follows.

$$p(x_{t-1}, y_{t-1}) = \mathcal{N}(x_{k-1}; \hat{x}_{k-1|k-1}, P_{k-1|k-1}) \tag{10}$$

$$p(x_t, y_{t-1}) = \mathcal{N}(x_k; \hat{x}_{k|k-1}, P_{k|k-1}) \tag{11}$$

$$p(x_t, y_t) = \mathcal{N}(x_k; \hat{x}_{k|k}, P_{k|k}) \tag{12}$$

For Kalman filter (4) can be rewritten as (13)

$$\hat{x}_{k-1|k-1}, P_{k-1|k-1} \xrightarrow{(Predict)} \hat{x}_{k|k-1}, P_{k|k-1} \xrightarrow{(Update)} \hat{x}_{k|k}, P_{k|k} \xrightarrow{(Predict)} \hat{x}_{k+1|k+1}, P_{k+1|k+1} \tag{13}$$

During this predict-update process Kalman filter tries to minimize the value between the predicted value and the measurement value by using a scaling parameter called Kalman gain. The whole flow of the Kalman filter can be summarized as Figure 2.2.

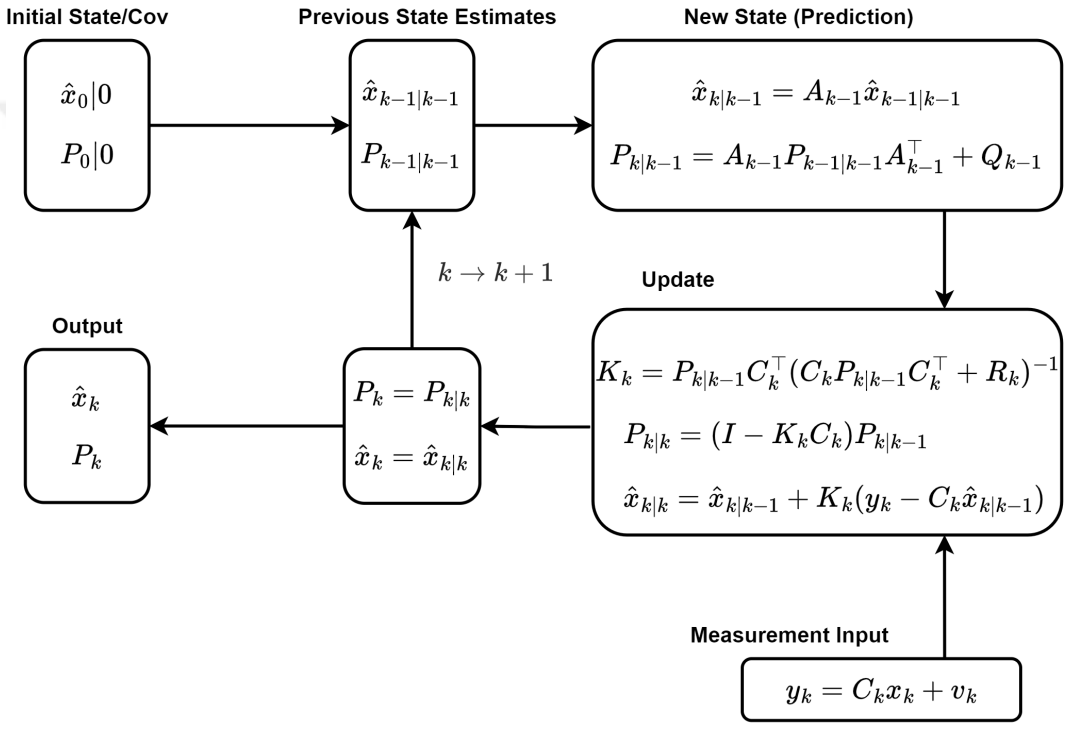

Figure 2.2 Kalman Filter Flow

y_k represents measurement input. For N sensor measurements, y_k can be expressed as a matrix of size Nx1.

When considering Kalman gain equation, noise covariance matrix P and measurement noise matrix R seem to be a very important for filter performance. When the system modeling is done properly, it is expected that the P value will gradually decrease and be limited to the Q value. At this point, if the error noise covariance is considerably smaller than the

measurement noise, the Kalman gain will approach zero. P and R values are critical to obtain a meaningful output from data from different sensors.

In general implementations of the Kalman filter, process noise and measurement noise are considered constant. When the measurement noise changes, the performance of the Kalman filter decreases. There are various methods in the literature to improve this situation. In the next section, information about these studies will be given.

2.2.1. Combining Sensor Data

For each sensor, we obtain an estimated value and its associated error covariance value as a result of the Kalman filter. According to the Minimum Variance Unbiased Estimator, the best choice among these covariance values is the one with the lowest value [20].

We can define the Kalman filter result of two different sensors as in the equation below (14).

$$s_1 = x + v_1 \quad \text{and} \quad s_2 = x + v_2 \tag{14}$$

x represents the estimated position, while v_1 and v_2 represent the error in this estimation. If we don't have any other information, we can combine the sensor results in a linear manner as seen in the equation (15) below.

$$\hat{x} = w_1 s_1 + w_2 s_2 \tag{15}$$

We can express the estimation error as in the following equation (16).

$$\widetilde{x} = \hat{x} - x \tag{16}$$

We can reach the optimal solution by minimizing the mean square value of estimation error $\widetilde{x}$. As seen in equation (17), optimal solution occurs when the expected value of the estimation

error is equal to 0.

$$\begin{aligned} E[\widetilde{x}] &= E[\hat{x} - x] = 0 \\ &= E[w_1 s_1 + w_2 s_2 - x] = 0 \\ &= E[w_1(x + v_1) + w_2(x + v_2) - x] = 0 \end{aligned} \tag{17}$$

Since $E[v_1] = E[v_2] = 0$, equation (17) can be arranged as in equation (18) below. In order for the below equation to be satisfied, $w_1 + w_2 = 1$.

$$E[(w_1 + w_2)(x)] - E[x] = 0 \tag{18}$$

The variance of the error can be found using the equation (19) below [21].

$$\begin{aligned} E[\widetilde{x}^2] &= E[(\hat{x} - x)^2] \\ &= E[(w_1 s_1 + w_2 s_2 - x)^2] \\ &= E[(w_1(x + v_1) + (1 - w_1)(x + v_2) - x)^2] \\ &= E[(w_1 x + w_1 v_1 + x - w_1 x + (1 - w_1)v_2 - x)^2] \\ &= E[(w_1 v_1 + (1 - w_1)v_2)^2] \\ &= w_1^2 E[v_1^2] + (1 - w_1)^2 E[v_2^2] + 2w_1(1 - w_1)E[v_1 v_2] \end{aligned} \tag{19}$$

Variance of v_1 and v_2 can be express $E[v_1^2] = \sigma_1^2$ $E[v_2^2] = \sigma_2^2$. Also, $E[v_1 v_2] = 0$ because v_1 and v_2 are uncorrelated with each other. Using this information, we can equation (19) can be arranged as in equation (20) below.

$$E[\widetilde{x}^2] = w_1^2 \sigma_1^2 + (1 - w_1)^2 \sigma_2^2 \tag{20}$$

We can find the expression for w_1 that minimizes this equation by taking the derivative of the equation and setting it equal to 0

$$\begin{aligned} (w_1^2 \sigma_1^2 + (1 - w_1)^2 \sigma_2^2)' &= 0 \\ 2w_1 \sigma_1^2 + 2(1 - w_1)\sigma_2^2 &= 0 \end{aligned} \tag{21}$$

and solve equation for w_1 we can obtain equation (22)

$$w_1 = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \quad \text{and} \quad w_2 = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \tag{22}$$

According to the result, the values w_1 and w_2 must always be between 0 and 1 which means that the variance rate will be better than if only σ_1 or σ_2 is used [22].

2.2.2. Maximum Correntropy Criterion Kalman Filter

When modeling the Kalman filter, it assumes that the process and measurement noise are Gaussian. However, in practical applications, this noise condition is not met and the noise from the environment is unavoidable. Unexpected noise interference like non-Gaussian noise or shot noise can corrupt our sensor data [23–25]. In this case, the performance of the Kalman filter begins to deteriorate. In cases of non-Gaussian noise and large outliers, a new method is proposed called maximum correntropy criterion Kalman filter (MCC-KF). This filter has same prediction-update procedure as the KF, only difference is the gain. Also MCC investigate higher order moments of information rather than Kalman filter second order information [26].

Correntropy can be defined as a statistical metric of the similarity between two random variables [27]. The cross-correntropy of two scalar random variables X and Y can be defined as equation (23)

$$C_z(X, Y) = E[k_z(X, Y)] = \iint k_z(x, y) f_{xy}(x, y)\, dx\, dy \tag{23}$$

Without knowledge of joint probability density of two random variable, we can calculate integral with finite N point as equation (24)

$$\hat{C}_z(X, Y) = \frac{1}{N} \sum_{i=1}^{N} k_z(x_i, y_i) \tag{24}$$

To investigate higher order moment k_z is chosen Gaussian kernel and equation (24) turns to equation (25).

$$\hat{C}_z(X,Y) = \frac{1}{N}\sum_{i=1}^{N} G_z(x_i - y_i) \Rightarrow k_z(x_i, y_i) = G_\sigma(x_i - y_i) = e^{(-\frac{||x_i - y_i||^2}{2z^2})} \tag{25}$$

$||.||$ represent L2 norm of vector and z is the user specified bandwidth or kernel size. As seen above equation (25) Gaussian kernel function bounded between 0 and 1. If $X = Y$ result will be maximum. The MCC-KF is derived from cost function (26)

$$J_m = G_z(||y_k - C\hat{x}_k||_{R_k^{-1}}) + G_z(||\hat{x}_k - A\hat{x}_{k-1}||_{P_{k|k-1}^{-1}}) \tag{26}$$

Minimizing cost function (27) taking derivative with respect to $\hat{x}_k$ and equalize to 0.

$$\begin{aligned}\frac{\partial J_m}{\partial \hat{x}} = 0 \Rightarrow -\frac{1}{z^2}G_z(||y_k - C\hat{x}_k||_{R_k^{-1}})C^T R_t^{-1}(y_k - C\hat{x}_k) + \\ \frac{1}{z^2}G_z(||\hat{x}_k - A\hat{x}_{k-1}||_{P_{k|k-1}^{-1}})P_{k|k-1}^{-1}(\hat{x}_k - A\hat{x}_{k-1})\end{aligned} \tag{27}$$

$$P_{k|k-1}^{-1}\hat{x}_k - P_{k|k-1}^{-1}A\hat{x}_{k-1} = L_k H^T R_k^{-1}(y_k - C\hat{x}_k) \tag{28}$$

$$L_k = \frac{G_z(||y_k - C\hat{x}_k||_{R_k^{-1}})}{G_z(||\hat{x}_k - A\hat{x}_{k-1}||_{P_{k|k-1}^{-1}})} \tag{29}$$

$$(P_{k|k-1}^{-1} + L_k H^T R_k^{-1} H)\hat{x}_k = P_{k|k-1}^{-1}\hat{x}_k^- + L_t H^T R_k^{-1} y_k \tag{30}$$

To obtain new state estimate adding and subtract $L_t H^T R_t^{-1}\hat{x}_t^-$ right side of equation (30).

$$(P_{k|k-1}^{-1} + L_k H^T R_k^{-1} H)\hat{x}_k = P_{k|k-1}^{-1}\hat{x}_k^- + L_k H^T R_k^{-1} y_k + L_k H^T R_k^{-1}\hat{x}_k^- - L_k H^T R_k^{-1}\hat{x}_k^- \tag{31}$$

$$(P_{k|k-1}^{-1} + L_k H^T R_k^{-1} H)\hat{x}_k = (P_{k|k-1}^{-1} + L_k H^T R_k^{-1} H)\hat{x}_k^- + L_k H^T R_k^{-1}(y_k - H\hat{x}_k^-) \tag{32}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - H\hat{x}_k^-) \tag{33}$$

$$K_k = (P_{k|k-1}^{-1} + L_k H^T R_k^{-1} H)^{-1} L_k H^T R_k^{-1} \tag{34}$$

As seen above equation (34) MCC-KF tries to improve estimation performance by changing the Kalman gain in case of non-Gaussian noise.

3. SIMULATION ENVIRONMENT

Studies have been carried out on the system identification of multisensory integration dynamics in zebrafish. Due to the positive rheotaxis observed in zebrafish, they protect their position by turning their bodies towards the direction of the water flow. Objects in the flowing water prevent water flow in certain regions, creating regions behind them with relatively lower flow velocities. Especially in cases where the current speed is high, the fish aim to swim behind these objects to avoid being dragged by the current. Uyanık *et al* proposed a novel experimental setup for examining this type of multisensory integration in zebrafish. In this experimental setup, a flow tunnel was created, and the water flow was provided by two motors connected from the outside. With a computer-controlled system, a transparent tube is positioned to move forward and backward on an axis perpendicular to the flow direction. The transparent tube creates a region with low flow velocity that stimulates fish's mechanoreceptors (lateral line organ) without generating any visual stimuli, while a model fish placed in the tube is used for the visual stimulus [28].

Based on this experimental setup, a robotic simulation environment was created using the Robot Operating System-based Gazebo simulator.

3.1. Robot Operating System (ROS)

Compared to known operating systems, ROS is not exactly an operating system but a flexible framework for developing software for robots and robotic systems [29]. It provides a set of tools, libraries, and packages for robotic platforms.

ROS architecture consists of nodes, messages, topics, and services [30]. These are controlled by the ROS Master. The ROS Master is actually a main node. The master prepares the system necessary for the nodes to work [31]. Nodes are individual executable units that perform specific tasks. For example, a node is used to collect or control sensor data. Nodes communicate with each other by using topics. We can define topics as communication channels that allow nodes to exchange data. There is an asynchronous publish-subscribe

mechanism between nodes and topics. Nodes can publish data to a topic, and other nodes can subscribe to that topic to receive the data. The content of the data exchange between nodes and topics is defined by messages. The ROS concept can be summarized as follows: Figure 3.1.

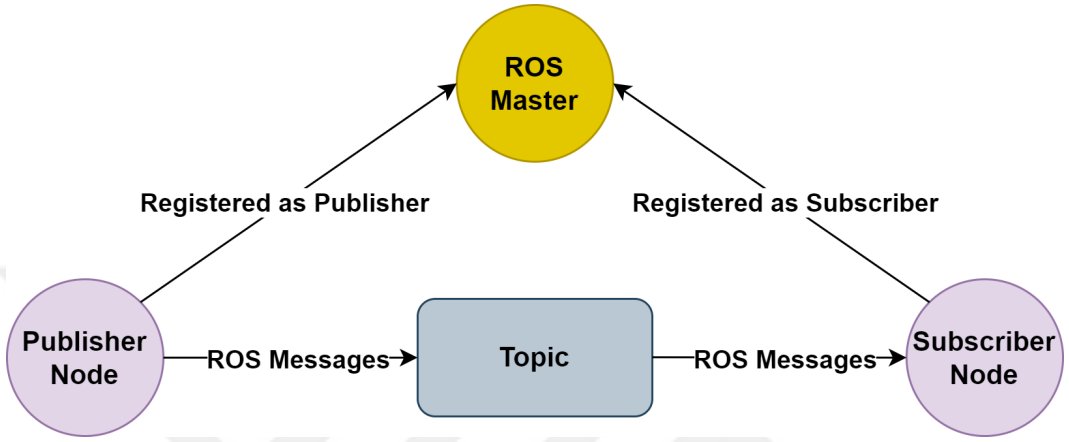

Figure 3.1 ROS Concepts

3.2. Gazebo

Gazebo is an open-source simulation toolbox utilized in robotics. By creating a virtual 3D, Gazebo facilitates robot simulations, offering the advantage of abstracting from hardware constraints to establish a versatile working environment [32]. It is both cost- and time-effective, and it allows for visualization from the robot's perspective [33]. Gazebo can work with ROS, allowing us to use many advantages of ROS. For instance, reading camera sensor data through ROS enables visualization of camera-recorded information in Gazebo. Moreover, this setup permits testing desired algorithms using diverse scenes within the camera data. Leveraging Gazebo's potential, we can define our desired physical environment using URDF (Universal Robot Description Format) files [34],[35].

3.3. Simulation Setup

A simulation environment was created using MATLAB, ROS, and Gazebo. The simulation environment's flow chart is shown in Figure 3.2 below.

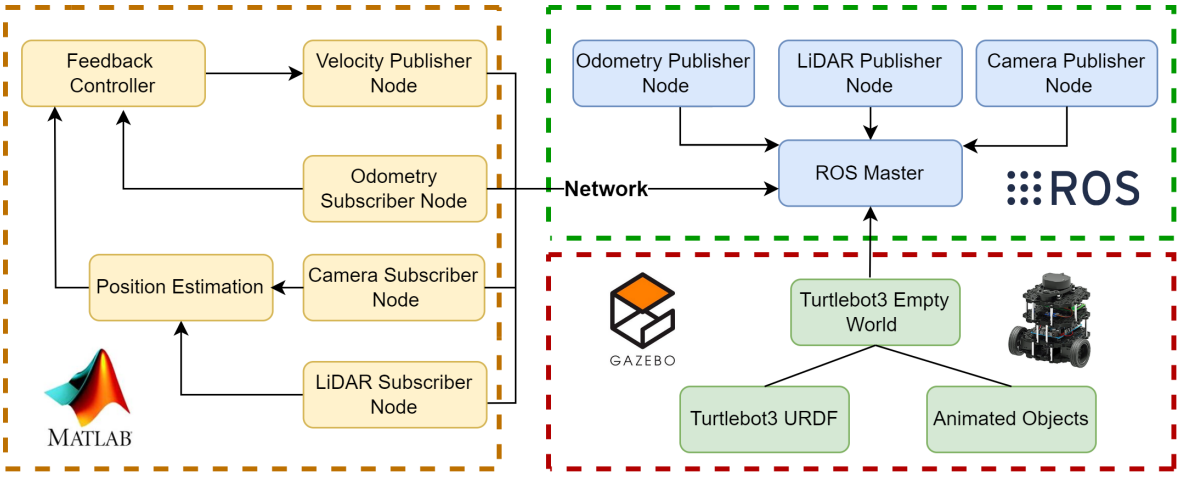

Figure 3.2 Flowchart of Simulation Setup

Before engaging with a physical system, a comprehensive evaluation of the software architecture and diverse algorithms were conducted by developing a single-axis simulation environment within the Gazebo platform. Drawing inspiration from the zebrafish setup, our configuration involves two objects that move along a singular axis. This setup incorporates LiDAR and camera sensors for precise object detection. Specifically, one object takes the form of a transparent, two-dimensional rectangular plate, while the other assumes the appearance of a red spherical entity.

The LiDAR sensor demonstrates an ability to detect the transparent object, whereas the camera sensor exclusively identifies the red sphere. The animation components have been written in C++ language, enabling the Gazebo-generated shapes to execute defined movements. Within this framework, the Turtlebot3 robot served as the foundation for sensor integration. Renowned for its extensive use in educational, research, and recreational contexts [36],[37], the Turtlebot3 operates on the ROS platform—a preferred choice among robotic enthusiasts. There is also a Gazebo model with all the physical features of Turtlebot3. This model boasts a comprehensive sensor suite, including components such as odometry, LiDAR, and a camera. Real-time data acquisition and sensor information processing became attainable by leveraging the capabilities of the Gazebo simulation environment.

The ROS Toolbox within MATLAB helps us leverage many functionalities offered by the ROS environment. Additionally, MATLAB enables the creation of a dedicated ROS master. This ROS master seamlessly communicates with other ROS masters across the network [38]. A network between MATLAB and Gazebo has been established within our simulations, facilitating the seamless data exchange through this network.

The simulation depicts the arrangement of target objects and Turtlebot3, as illustrated in Figure 3.3 below. The Turtlebot3 features a front-facing Camera and LiDAR sensors. To enable unidirectional movement for Turtlebot3, alterations were implemented in the URDF file,

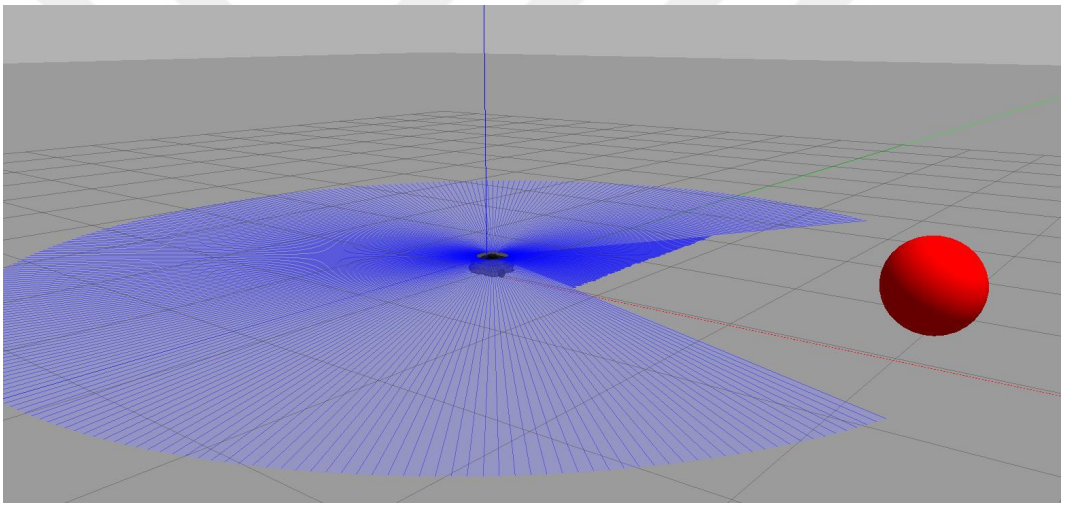

Figure 3.3 Gazebo Simulation Environment

Blue lines show the area scanned by LiDAR. While LiDAR can only detect the transparent object in front of it, it cannot detect the other object. We can extract the midpoint of the moving object by averaging the data obtained from LiDAR. MATLAB also allows visualizing LiDAR data. In this way, we can detect unwanted data more efficiently. Visualized LiDAR data looks like in Figure 5. X and Y show the coordinates of simulation planes.

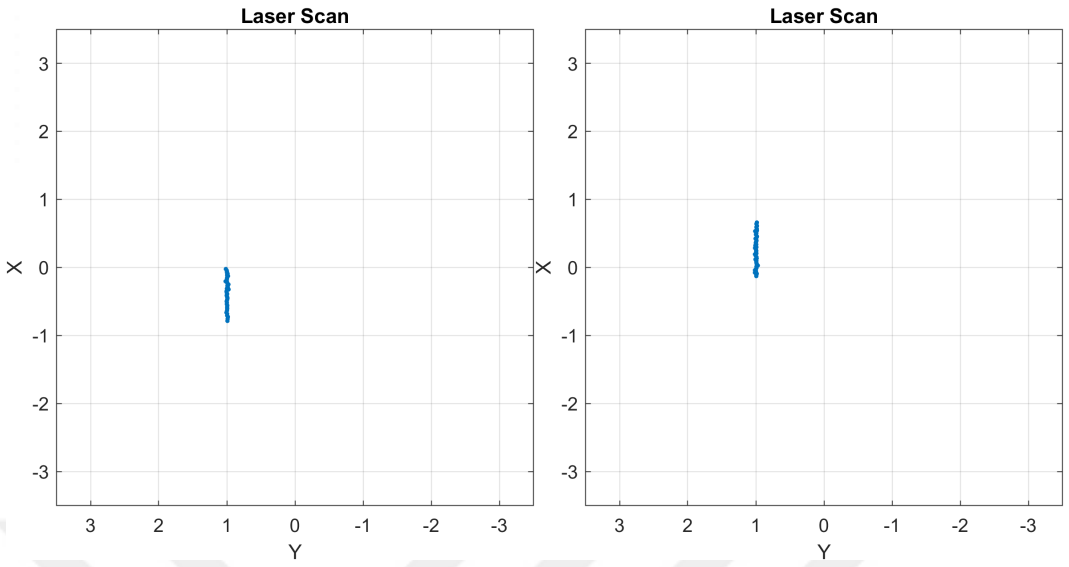

Figure 3.4 Visualized LiDAR Data

The camera mounted on Turtlebot 3 captures images while tracking the red sphere's movement. These acquired images are first converted to grayscale and subsequently binarized. Calculating the average of the binarized points yields the center of the moving object. The visualized version of camera images with the determined center point in MATLAB has been shown in Figure 3.5 below.

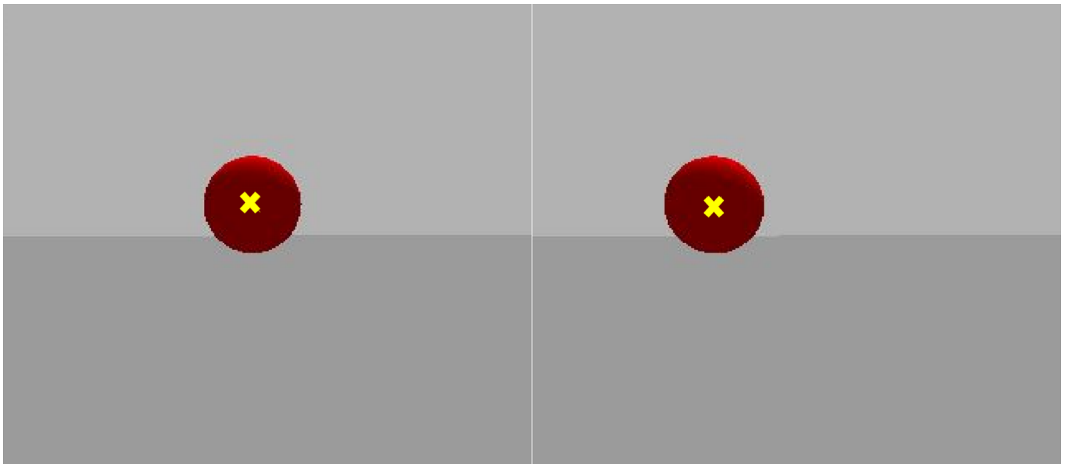

Figure 3.5 Visualized Camera Data

In MATLAB, subscriber nodes are established to receive data from LiDAR and camera sensors at 10 Hz. Beyond these two sources, we can also extract the robot's position

information during simulation by utilizing Turtlebot3's odometry data. A dedicated node subscribing to the odometry data has been created to achieve this.

The feedback controller diagram is given below Figure 3.6. We computed the average location data acquired from LiDAR and the camera. We derived an error signal by subtracting the robot's positional data from this information. This error signal served as the input for a feedback controller comprising a PID and a plant. The signal generated at the output of the PID is transmitted to Turtlebot3 as velocity information; the plant performed the robot's movement with the velocity information it received, and the odometry data was taken at the output and compared with the reference signal.

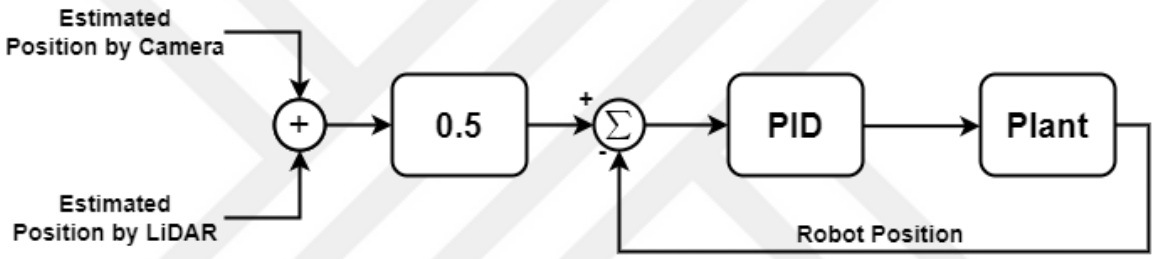

Figure 3.6 Feedback System of Simulation Setup

The motion of target objects is expressed as the sum of sinusoidal signals with two different amplitudes and frequencies as in equation (35).

$$(0.3)(sin(2\pi 0.05t)) + (0.1)(sin(2\pi 0.1t)) \tag{35}$$

The time and frequency domain results of the experiment performed by taking the average of two sensor data are given in Figure 3.7. We observed that the tracking performance is good when the position estimation is accurately derived from sensor data. Additionally, the integration of the data with a suitable sensor fusion method further enhances this performance.

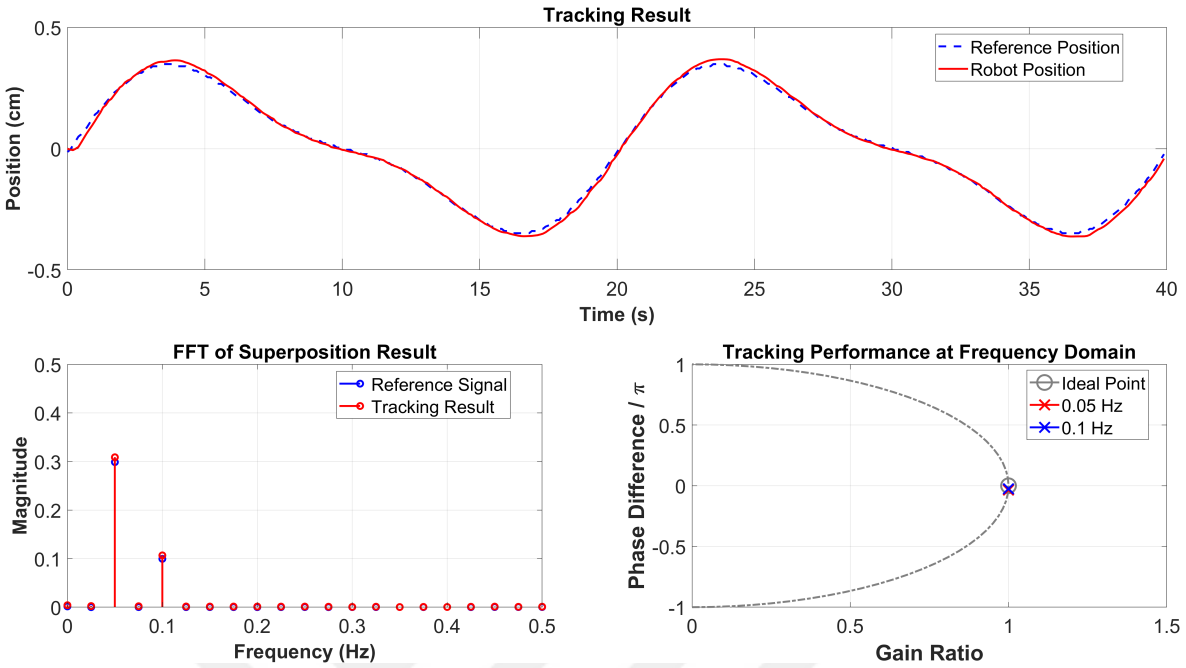

Figure 3.7 Simulation Result

Following the simulation results here, we had the opportunity to perform the software infrastructure and preliminary tests of our work on the physical setup. An example of the algorithm studies mentioned in the subsequent sections has also been executed on the simulation setup presented here.

4. EXPERIMENTAL SETUP

4.1. Mechanical Structure

Based on this experimental setup, we also created a novel experimental setup to examine sensor fusion in robotic system , as shown in Figure 4.1. In this setup, 3 different platforms are moving on a single axis. One of these platforms contains the target object to be followed by the camera, one includes the object to be followed by LiDAR, and the other contains the platform where the camera and LiDAR sensor are located. There is one stepper motor on the axis where each platform is located. A gear is placed on the stepper motor moving apparatus. At the opposite end of the axis where the stepper motor is located, another gear is attached to a metal bar. There are holes in the bottom of the platform, and a rope is tied through these holes. Then, this rope is attached to the gears at both ends of the axis. This way, the platform movement can be accomplished with the commands sent to the stepper motor. To detect the target stimuli accurately, the target object of the camera is chosen as a green circle, and the target object of the LiDAR is chosen as a long thin object that does not affect the image taken by the camera. In addition, since the camera and the platform on which the LiDAR is located will move during target tracking, another circular object has been placed in front of the camera to estimate the positions of the target objects accurately. As a result, the relative positions of the platforms can be estimated during target tracking.

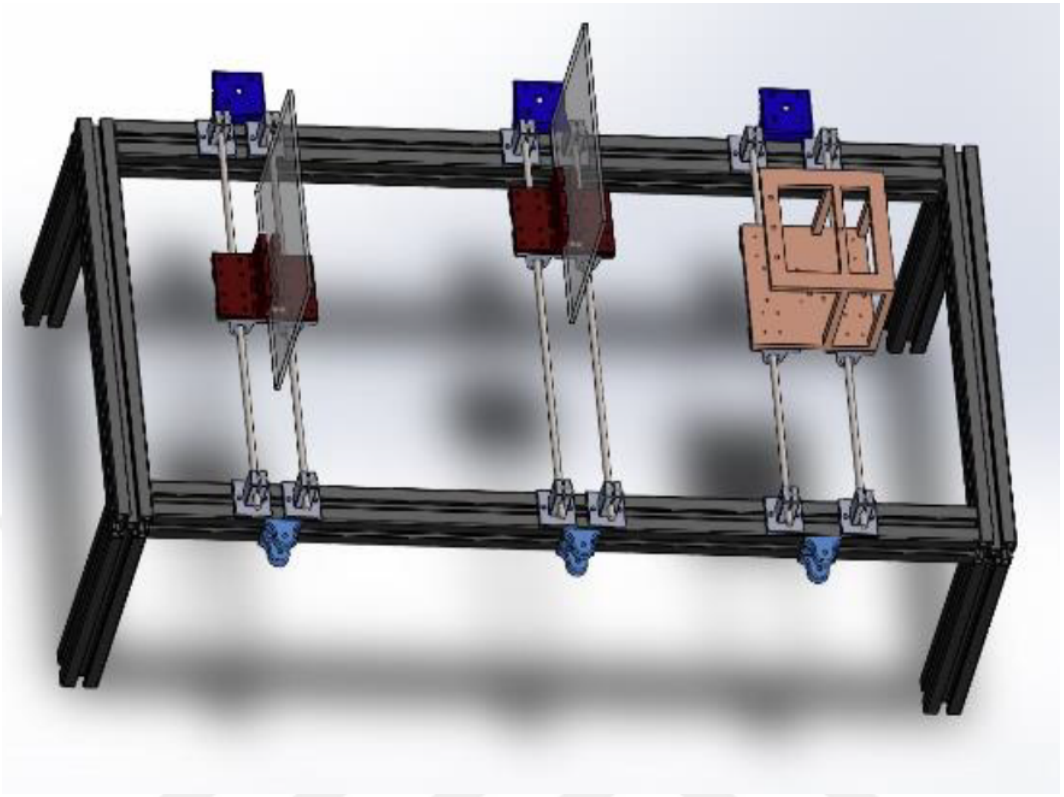

Figure 4.1 Mechanical Design of Experimental Setup

4.2. Tracking Task

In the experimental setup shown in Figure 4.1, a separate target object was placed for each sensor to detect movement. Both target objects performed the same movement during the experiments, starting from the same point. The sensor mechanism actually detects this as a single movement. The reason for choosing two different targets is to be able to examine the independent movements in future studies .

During tracking, the difference between the movement of the platform on which the sensors are located and the movement of the target objects tries to equalize to zero.

4.3. Electrical Interface

The electrical connections of the experimental setup are shown in Figure 4.2. In the system, 3 Nema 17 bipolar stepper motors are used to provide the movement of the target objects

and the sensor system. The motors are connected to a 12V power supply. This voltage is used to generate the power required for the rotation of the motors. TMC5160 motor driver is used to perform the motor movements in a controlled way. TMC5160 has an SPI interface. Communication with Arduino Mega is provided using this interface. The 5V power requirement of the communication lines in the TMC5160 is met from the outputs on the Arduino. Hokuyo URG-04-LX is used as a LiDAR sensor. This sensor can receive data at 10Hz with a resolution of 0.36 degrees in an angular field of 240 degrees. LiDAR works with 5V, and the power supply provides this. Data is transferred to the computer with the UART interface of LiDAR. Logitech C922 web camera is used to visually detect the movement of the target object. This camera can take images in 1080p/30 fps resolution. The computer provides the power required for the camera to work and the transfer of camera data via the USB connection. All electronic equipment used in the system is controlled by a computer.

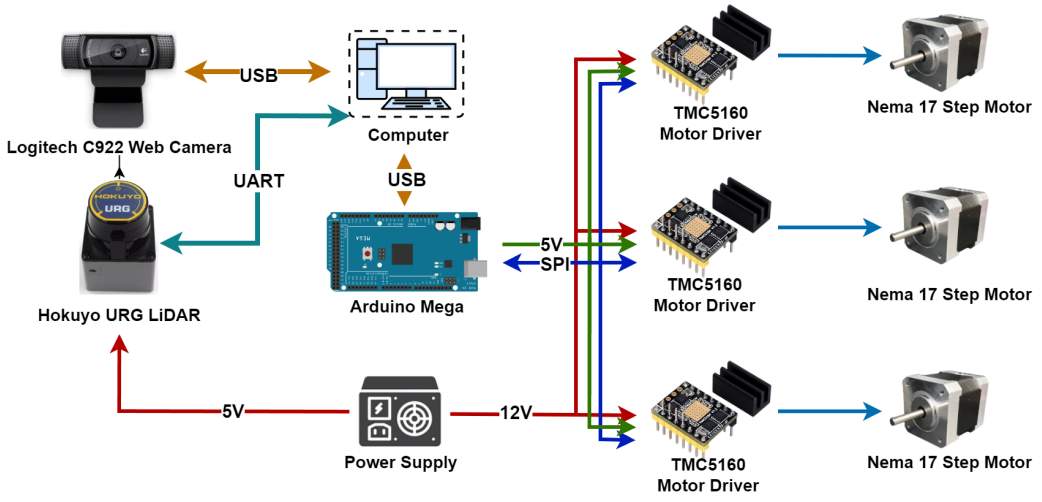

Figure 4.2 Electrical Interface Diagram

4.4. Software Architecture of the Experimental Setup

The basis of the software architecture of the experimental setup is ROS (Robot Operating System), a Linux-based operating system running on the computer. It is possible to use programs that can run simultaneously in parallel with each other on this operating system.

In this direction, two sub-modules were created that read the data from the camera and LiDAR. Each submodule uses 'topics' in ROS to read data from sensors. Topics contain specialized information messages from sensors. After the received messages, it goes through the processes mentioned in the previous sections for position estimation. Messages broadcast from both sensors are collected with the help of a subscriber. The collected data goes through sensor fusion processes and a single position estimation is generated. The result information is sent to the microcontroller via UART. The PI controller in the microcontroller generates the information necessary to drive the stepper motors and send them to the motors. All these processes are summarized in Figure 4.3.

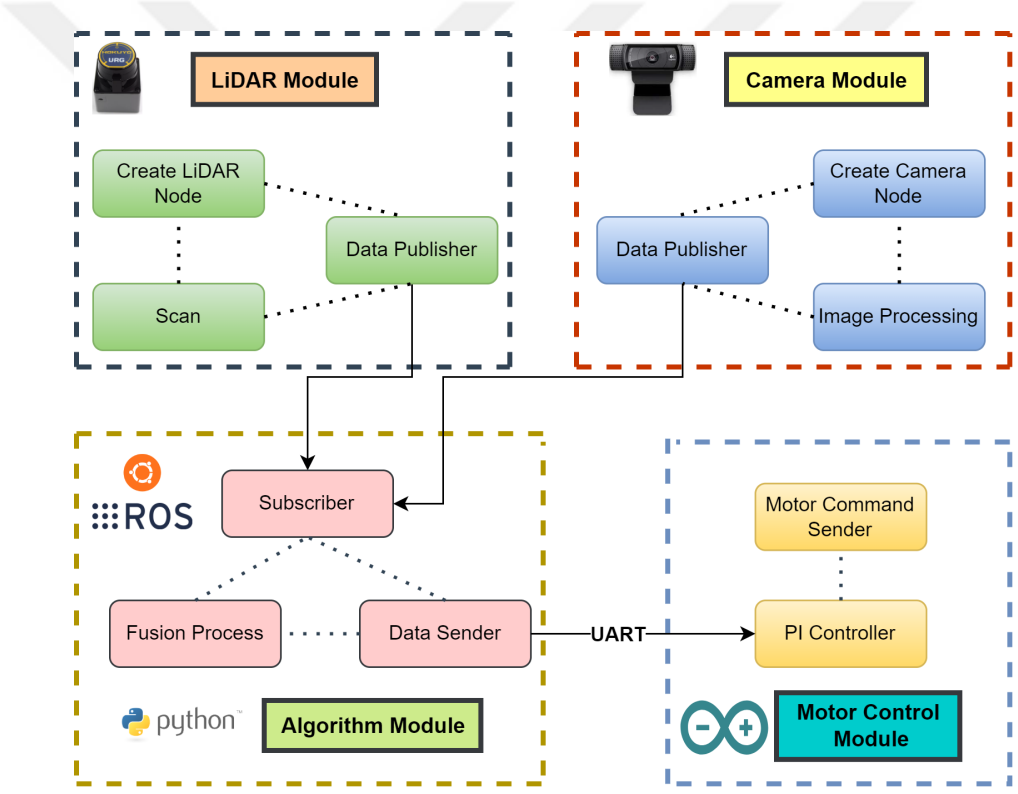

Figure 4.3 Software Architecture

4.5. Collection of Sensor Data

We need to track the targets's position estimation. Camera and LiDAR can not provide this information directly. For this reason, camera and LiDAR data have to go through a process. This section explains in detail what processes the data go through.

4.5.1. Position Estimation from Camera Data

There are two circular objects (one on the moving target object and the other fixed on the wall) in front of the camera in the system. Our aim is to find the pixel value that the midpoint of these two circular objects corresponds to in the frame. The procedures performed for this purpose are shown in the Figure 4.4.

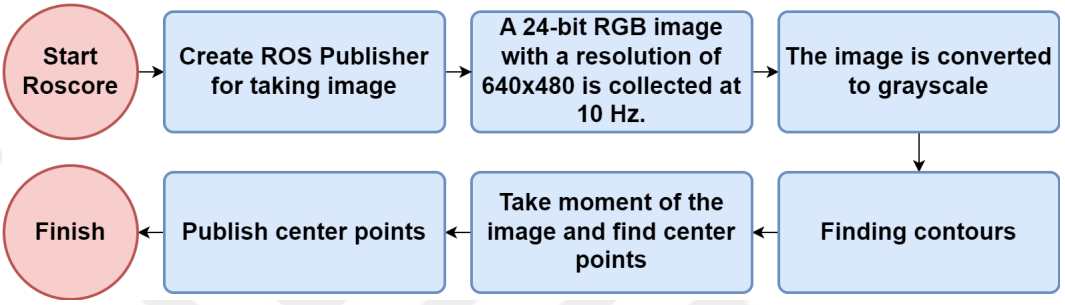

Figure 4.4 Image Processing Algorithm

With this algorithm, we try to find the horizontal pixel value of circular object's center. The operations in the algorithm steps were performed using the OpenCV library. The processing steps on the raw image are shown in Figure 4.5. The center point of the circle in the whole frame in the moving mechanism is interpreted as the position of the target object. The midpoint of the circle on the wall is used to estimate the positions of the target objects while the sensor platform is in motion.

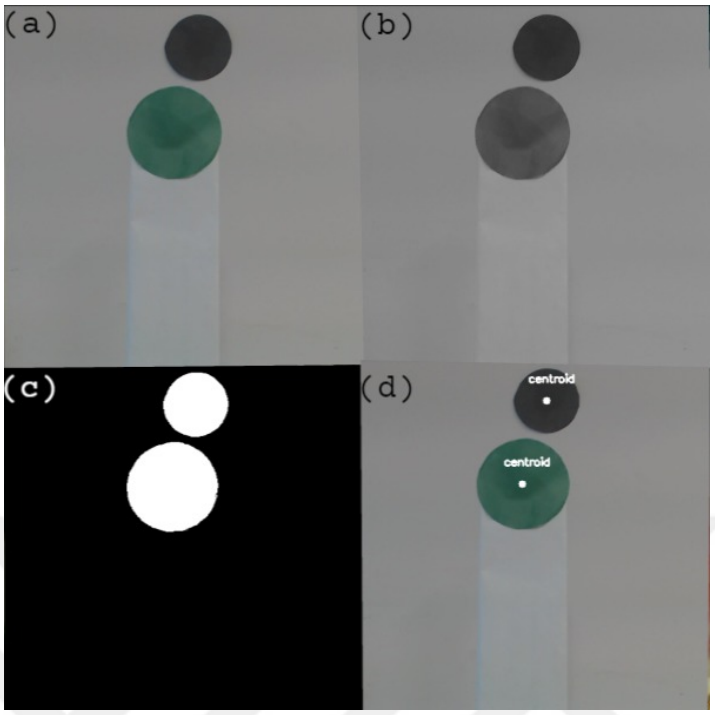

Figure 4.5 Image Processing Algorithm Steps, (a) raw image, (b) grayscale image, (c) threshold result, (d) showing center point of object

4.5.2. Position Estimation from LiDAR Data

To collect data using LiDAR, a long thin object was placed 20cm-25cm in front of the LiDAR. We can visualize this object detected by LiDAR using ROS's 3D visualization tool rviz, as shown in Figure 4.6. The midpoint of the image represents the location of the sensor assembly, the gray areas represent the scene the assembly is in, and the red dots represent the object detected by LiDAR.

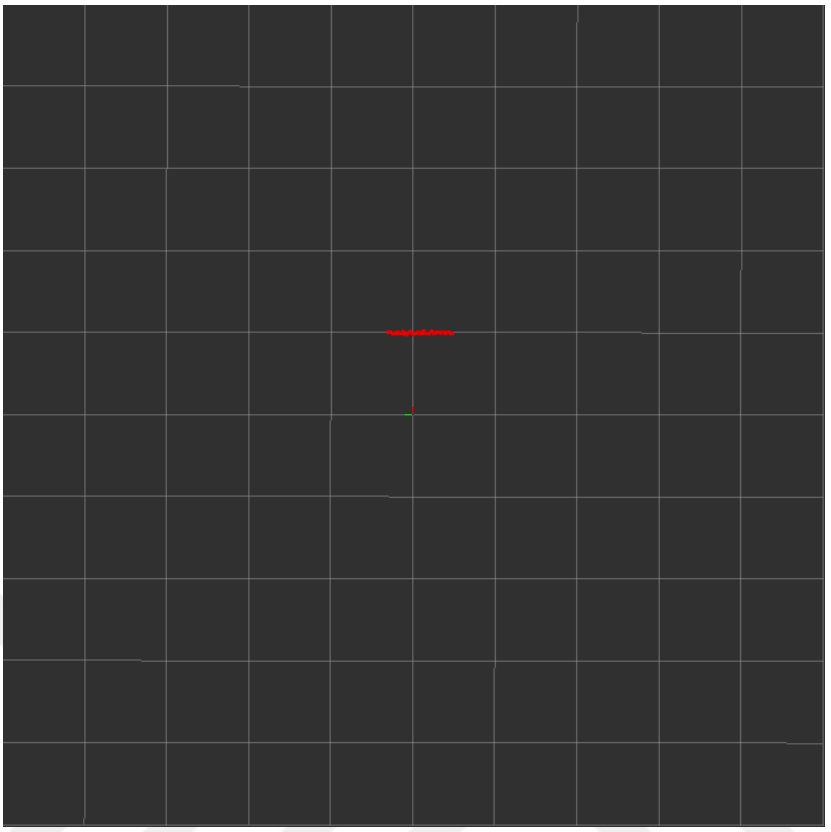

Figure 4.6 Visualization of the Object Detected by LiDAR

The LiDAR algorithm tries to estimate the motion of the midpoint of this object. The procedures performed for this purpose are shown in the Figure 4.7.

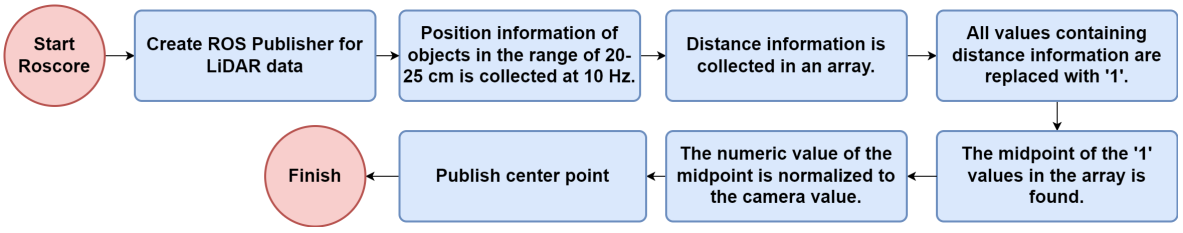

Figure 4.7 LiDAR Position Estimation Algorithm

The scanning area of the LiDAR is constrained to ensure accurate detection of the target object. To achieve this, the region in front of the LiDAR sensor was scanned within a 120-degree range, and the publisher node was configured to exclusively identify objects between 0.05 cm and 0.25 cm in size.

LiDAR sensor actually gives the distance information. While doing this, it keeps the distance information of the objects in the scanned area in an array. If the object cannot be detected in the scanned area, the values in the array are expressed as 0. The values in the array during the movement of the target object are shown in Figure 4.8.

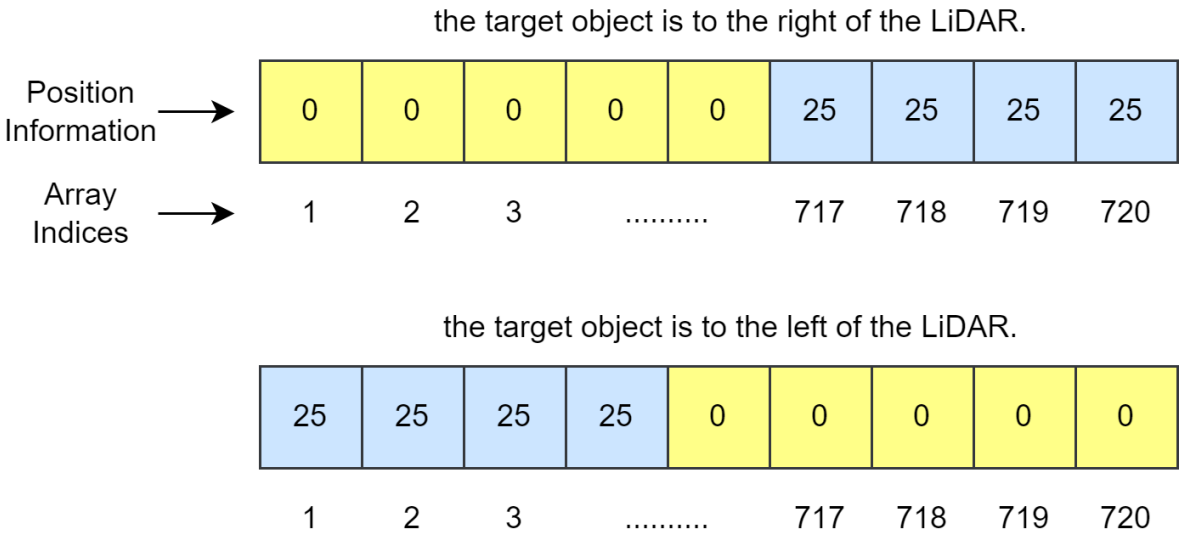

Figure 4.8 LiDAR Position Estimation Algorithm

Since the distance information of the target object is known, we changed the distance information in the array with the value 1. After this point, the median of the ones in the array gives us the midpoint of the object. We use the position of this midpoint in the array as the position of the object on the system. The numerical data obtained at the end of this process does not coincide with the camera information. The numerical data obtained from this process does not align with the camera information. Therefore, the LiDAR data is normalized according to the camera data. After normalization, the data resulting from object movements is interpreted uniformly by both the LiDAR and the camera.

5. SYSTEM MODEL

5.1. Closed-Loop System Model

In our system, a closed-loop system model has been used to combine sensor data and achieve successful tracking. It has been developed taking inspiration from Roth *et al* [39, 40] studies. The model used is as shown below Figure (5.1)

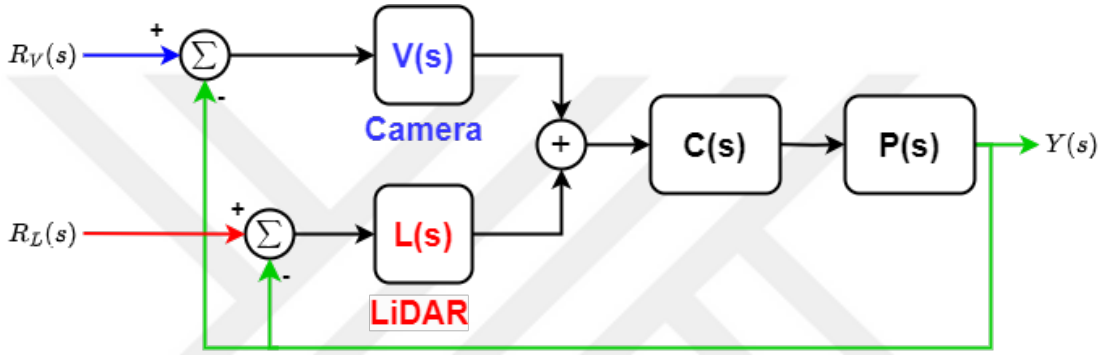

Figure 5.1 Multisensory Closed-Loop Model

All signals in the model are shown in the frequency domain. $R_V(s)$ represents visual target movements intended to be detected by the camera, and $R_L(s)$ represents reference target movements intended to be detected by LiDAR. $Y(s)$ indicates the location of the unit on which the sensors are located. $C(s)$ contains the PI controller that will drive the stepper motors. $P(s)$ represents the step motor plant. $V(s)$ represents the weight of the sensor data obtained from the camera, $L(s)$ represents the weight of the sensor data obtained by the LiDAR. We examine how the sensor data are combined, trying to estimate the values of $V(s)$ and $L(s)$ with precision.

We can express the transfer function of the model presented in the block diagram as follows.

$$[(R_V(s) - Y(s))V(s) + (R_L(s) - Y(s))L(s)]C(s)P(s) = Y(s) \tag{36}$$

$$C(s)P(s)[(R_V(s)V(s) + R_L(s)L(s)] = Y(s)[1 + C(s)P(s)(V(s) + L(s))] \tag{37}$$

$$G(s) = \frac{C(s)P(s)}{1 + C(s)P(s)(V(s) + L(s))} \tag{38}$$

Considering equation (37) and equation (38), we can rearrange the block diagram as Figure 5.2.

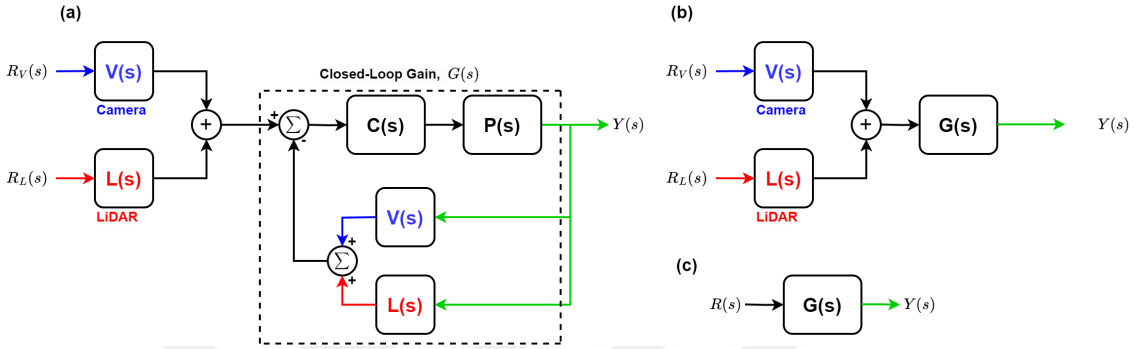

Figure 5.2 Rearrange of Multisensory Closed-Loop Model

If we choose $R_V(s) = R_L(s) = R(s)$ and $V(s) + L(s) = 1$, block diagram will be represent as Figure 5.2 (c).

The sensors in the robotic unit are able to measure the relative movement between their own movements and these reference movements. In this case, the two sensors make two different relative motion estimations based on their dynamics, noise levels, and sensitivity. The robotic system thinks that the information from two different sensors belongs to the same reference system, combines these two pieces of information, and makes a single relative motion estimation. Later, this estimation is used as the error signal to be taken to zero by the control unit .

5.2. Constant Velocity Kalman Filter Model

In the experimental setup, the target objects perform their movements in a sum of sine signal characteristics. The movement of target objects takes place very slowly. Based on this situation, a model that assumes the system speed as a constant is used in cases where the Kalman filter is applied. We assigned a small value to the Q matrix so that the Kalman filter

can accurately estimate since the target object motions do not move at a strictly constant velocity. The state transition matrix and measurement matrix we use in the constant velocity Kalman filter model [41] are as follows.

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{39}$$

The first two rows of matrix A are used for data from LiDAR sensors, and the last two rows are used for data from cameras. Thanks to this form of matrices, we can obtain estimation results for two sensors separately. The measurement results obtained from the sensors can be evaluated separately in the C matrix. The process noise covariance matrix form used in this model is as follows.

$$Q = \begin{bmatrix} V(x) & COV(x,v) & 0 & 0 \\ COV(v,x) & V(v) & 0 & 0 \\ 0 & 0 & V(x) & COV(x,v) \\ 0 & 0 & COV(v,x) & V(v) \end{bmatrix} \tag{40}$$

In the first two rows of the Q matrix, V(x) shows the variance of the LiDAR's position data, V(v) the variance of the velocity information in the data from the LiDAR, COV(x,v) and COV(v,x) the correlation between velocity and position. The values in the last two rows of the Q matrix represent the same relationships for the camera. We derived the matrix elements as follows [42].

$$\begin{aligned} V(x) = \sigma_x^2 &= E(x^2) - \mu_x^2 = E\left(\left(\frac{1}{2}a\Delta t^2\right)^2\right) - \left(\frac{1}{2}\mu_a \Delta t^2\right)^2 \\ &= \frac{\Delta t^4}{4}(E(a^2) - E(\mu_a^2)) = \frac{\Delta t^4}{4}\sigma_a^2 \end{aligned} \tag{41}$$

$$
\begin{aligned}
COV(x,v) = COV(v,x) &= E(xv) - \mu_x\mu_v \\
&= \left(\frac{1}{2}a\Delta t^2 a\Delta t\right) - \left(\frac{1}{2}\mu_a\Delta t^2\mu_a\Delta t\right) \\
&= \frac{\Delta t^3}{2}(E(a^2) - E(\mu_a^2)) = \frac{\Delta t^3}{2}\sigma_a^2
\end{aligned} \tag{42}
$$

$$
\begin{aligned}
V(x) = \sigma_v^2 &= E(v^2) - \mu_v^2 = E\big((a\Delta t)^2\big) - (\mu_a\Delta t)^2 \\
&= \Delta t^2(E(a^2) - E(\mu_a^2)) = \Delta t^2\sigma_a^2
\end{aligned} \tag{43}
$$

In the above equations, a represents the acceleration, σ_a^2 represents the variance of the acceleration, and Δt represents the sampling interval. Considering the equation results, we can express the Q matrix as follows.

$$
Q = \sigma_a^2 \begin{bmatrix}
\frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & 0 & 0 \\
\frac{\Delta t^3}{2} & \Delta t^2 & 0 & 0 \\
0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\
0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2
\end{bmatrix} \tag{44}
$$

Measurement noise of constant velocity model kalman filter represent with R matrix. This R matrix can be expressed as follows.

$$
R = \begin{bmatrix}
\sigma_{x_l}^2 & \sigma_{x_l}\sigma_{x_c} \\
\sigma_{x_c}\sigma_{x_l} & \sigma_{x_c}^2
\end{bmatrix} \tag{45}
$$

$\sigma_{x_l}^2$ represent variance of LiDAR position measurement. $\sigma_{x_c}^2$ represent variance of camera position measurement. Since there is no correlation between the measured data from these two sensors, we can simplify the equation (45) as follows.

$$
R = \begin{bmatrix}
\sigma_{x_l}^2 & 0 \\
0 & \sigma_{x_c}^2
\end{bmatrix} \tag{46}
$$

y_k contain measurements of LiDAR and camera sensors. This y_k matrix can be expressed as follows.

$$y_k = \begin{bmatrix} M_{L_{pos}} \\ M_{C_{pos}} \end{bmatrix} \tag{47}$$

These (39), (44), (46), (47) matrices were used in Kalman filter applications in experiments.

6. EXPERIMENTAL RESULTS

6.1. Weighted Average Based Sensor Fusion

In the first step of the experiments, tracking was tried to be done using a weighted average. For this, $V(s) = L(s) = 0.5$ was chosen in the diagram shown in Figure 5.2. In the scenario where the sensor data is not corrupted by noise, the tracking performance is monitored. Target objects $r_v(t) = r_l(t)$ perform the same motion. The equation (48) below describes this motion.

$$r_v(t) = r_l(t) = A_r((cos(2\pi 0.1) + \phi_{r1}) + (cos(2\pi 0.2) + \phi_{r2})) \tag{48}$$

In the ideal scenario, the system's response should be as in the following equation (49).

$$y(t) = A_y((cos(2\pi 0.1) + \phi_{y1}) + (cos(2\pi 0.2) + \phi_{y2})) \tag{49}$$

In the case of successful tracking of target objects, the gain and phase relationship should be observed as in the following equation (50).

$$Gain = A_y/A_r = 1, \qquad Phase = \phi_{r1} - \phi_{y1} = \phi_{r2} - \phi_{y2} = 0 \tag{50}$$

The RMSE value between the reference signal and the system response was checked to measure the tracking performance. RMSE is a concept that expresses the difference between the actual value and the predicted value in statistics. The smaller the RMSE value, the more successful the results. The RMSE is calculated as seen in equation (51).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}, \text{(N = number of data)} \tag{51}$$

The first experiment was carried out using all this information, and the results in Figure 6.1 were obtained. When we examine the results, we can see that the tracking process in the time

domain is very successful. There is a difference of approximately 0.2-0.3 seconds between the system response and the target objects. This difference is due to the time it takes to process the data and then generate the appropriate instruction by the motors. The sensor platform was able to track the movement of target objects properly. If we examine the FFT results, it is seen that the frequency components containing the motion of the target object exactly match the system response. We also notice that the relationship between gain and phase is quite close to the ideal point. These test results show that tracking can be performed successfully if the sensor data is reliable.

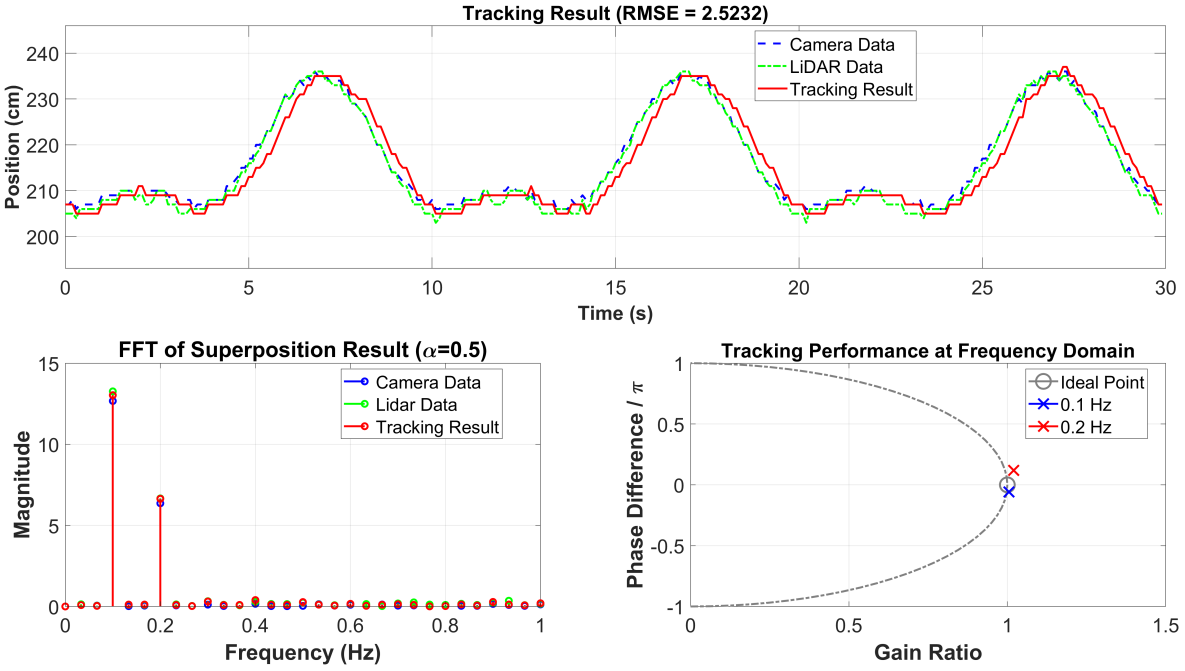

Figure 6.1 Weighted Average Based Sensor Fusion Result

6.2. Kalman Filter Based Sensor Fusion

Just as in a weighted average-based sensor fusion step, the reference signal containing the movement of target objects was not corrupted with noise in the first experiments of the constant velocity Kalman filter. The model used for the Kalman filter is given in equation (39). Since the errors resulting from the measurements of the sensors are very small, these values were not used as measurement noise and were ignored. Process noise and

measurement values are selected as in the following equation (52).

$$Q = \begin{bmatrix} 0.0025 & 0.05 & 0 & 0 \\ 0.05 & 1 & 0 & 0 \\ 0 & 0 & 0.0025 & 0.05 \\ 0 & 0 & 0.05 & 1 \end{bmatrix}, R = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \tag{52}$$

The position and velocity information of the two sensors are collected in a single matrix. After the Kalman filter process, we have an estimation and an error covariance of this estimation value. In the fusion step, the position estimation values are combined using inverse variance weighted. All these steps are defined in Algorithm 1.

Algorithm 1 Kalman Filter

Initialize:

$$x_0 = \begin{bmatrix} L_{pos} \\ L_{vel} \\ C_{pos} \\ C_{vel} \end{bmatrix}, P_0 = \begin{bmatrix} P_{L_{pos}} & 0 & 0 & 0 \\ 0 & P_{L_{vel}} & 0 & 0 \\ 0 & 0 & P_{C_{pos}} & 0 \\ 0 & 0 & 0 & P_{C_{vel}} \end{bmatrix}$$

for $k \leftarrow 1$ to N **do**

Prediction:

$x_k = Ax_{k-1}$

$P_k = AP_{k-1}A^T + Q$

Update:

$S = CP_{k-1}C^T + R$

$K = PCS^{-1}$

$x_{k+1} = x_k + K(y_k - x_k)$

$P_{k+1} = (1 - KC)P_k(1 - KC)^T + KRK^T$

Fusion:

$T_{pos} = \frac{P_{C_{pos}} L_{pos}}{P_{L_{pos}} + P_{C_{pos}}} + \frac{P_{L_{pos}} C_{pos}}{P_{L_{pos}} + P_{C_{pos}}}$

end for

If our model is correct in an ideal Kalman filter, the process noise should be 0 and the Kalman gain should converge to 0. But we have updated the Q matrix knowing there is a margin of error in our model, so the Kalman gain will not converge to 0. We will get the result by giving more weight to the estimated value between the estimated value and the measured value. When we examine the time and frequency domain results, we see that our Kalman filter model gives an accurate result. The RMSE value is very close to the case when we use the weighted average method.

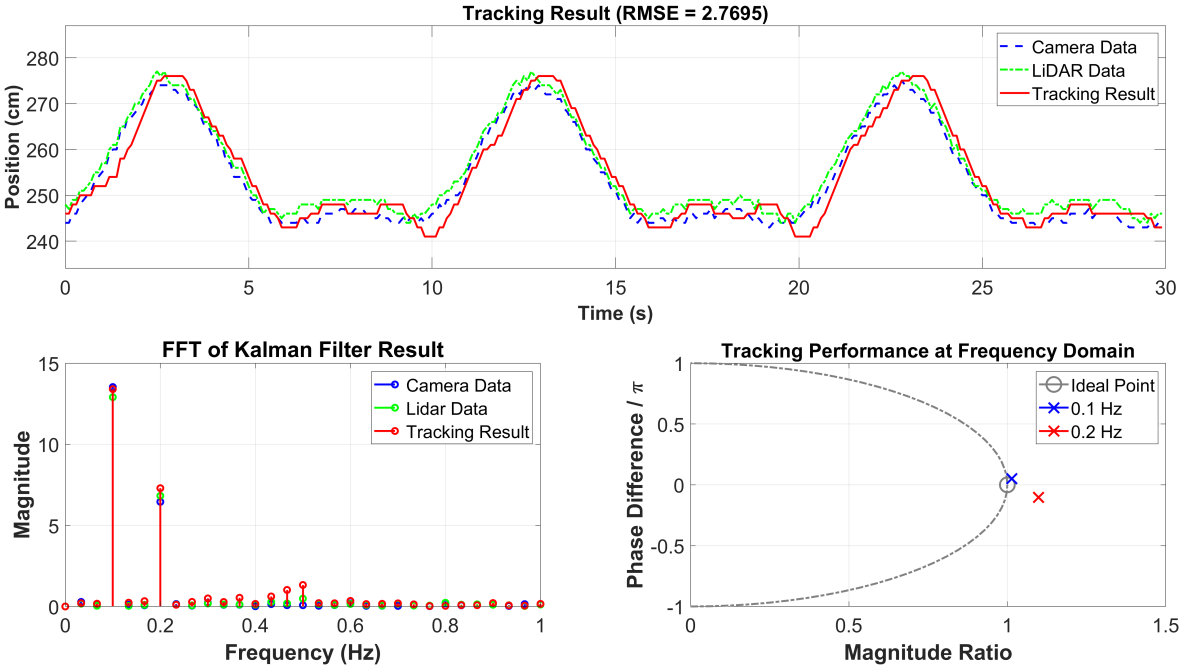

Figure 6.2 Constant Velocity Kalman Filter Model Result without Noise

The performance of the Kalman filter was evaluated by observing two parameters. In the first, the position error should be in the range of $\pm 3\sigma$, since it represents the range in which a value with a Gaussian distribution can be found 99.73%. Since the sensor data are close to each other and there is no noise in the environment, the error covariance calculated from the Kalman filter is equal for the two sensor results.

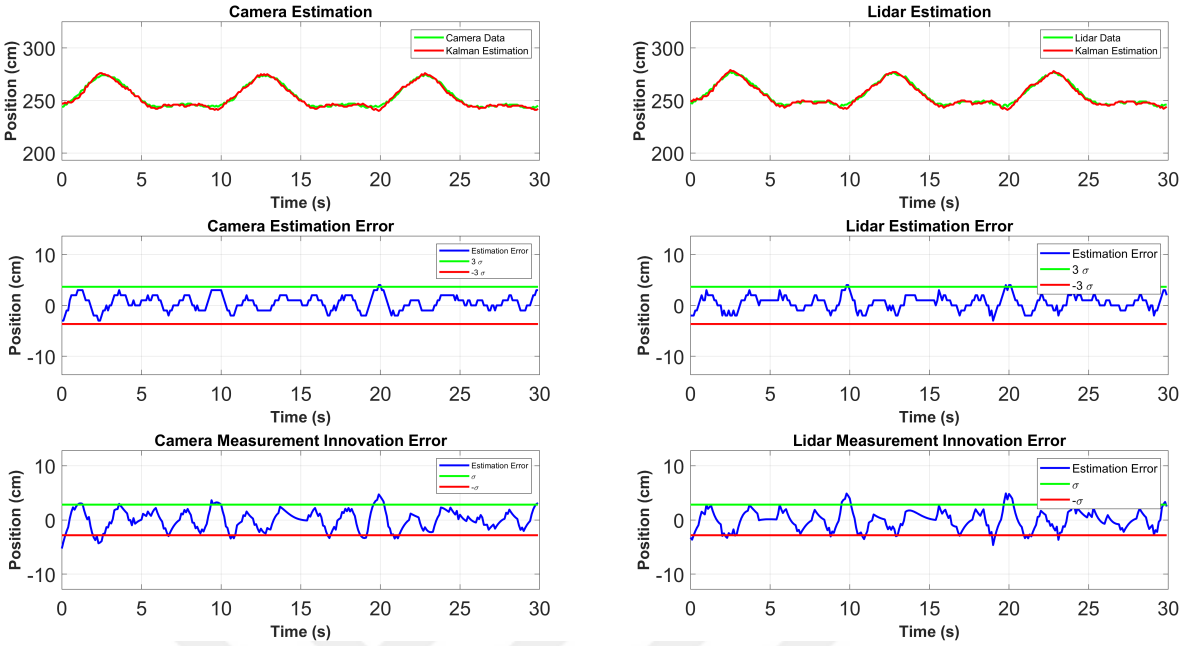

Figure 6.3 Constant Velocity Kalman Filter Model Performance without Noise

6.3. Kalman Filter Based Sensor Fusion with Residual Based Measurement Noise Estimation (Moving Window)

Under ideal conditions, receiving completely noise-free sensor data would not be possible. If we know about measurement noise, the assumption we made in the R matrix of the Kalman filter may only sometimes be correct. And in this case, it affects the performance of the Kalman filter. In the experiment in Figure 6.4, unlike the other experiment, the LiDAR data was corrupted by Gaussian noise after a certain period. This value is above the value we predicted in the R matrix. As can be seen in the time domain graph of 6.4, it is seen that there is corruption in both the tracking behavior and the LiDAR data after the 15th second.

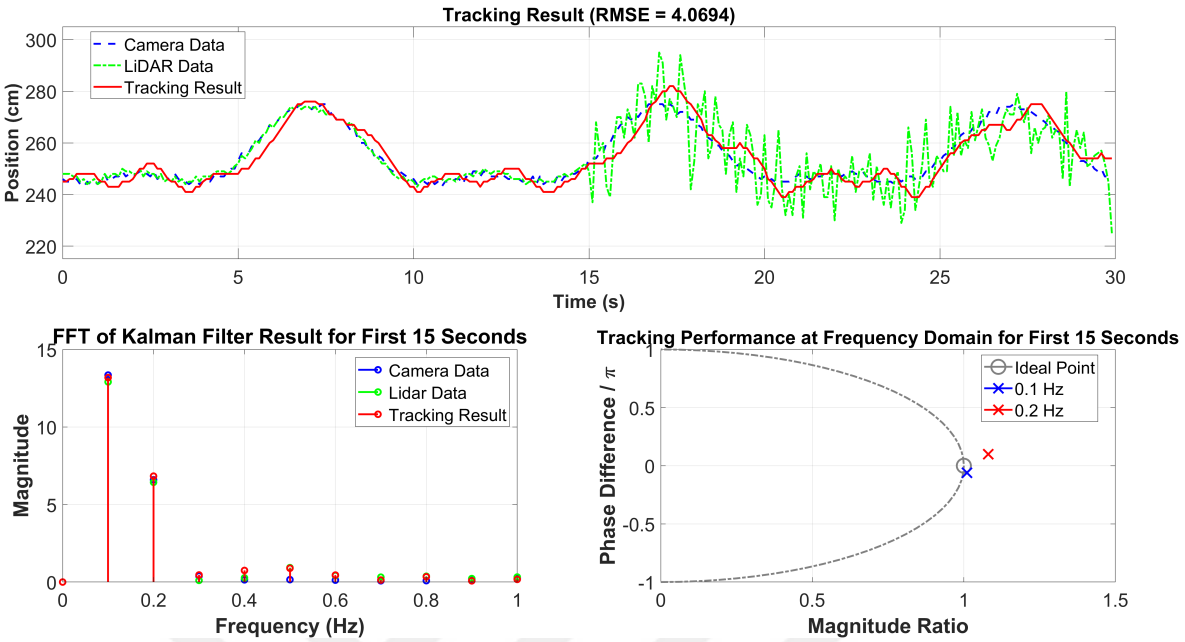

Figure 6.4 Constant Velocity Kalman Filter Model Result with Increase Noise

Figure 6.5 contains the FFT result of the last 15 seconds' data. In the LiDAR sensor, signals at different frequencies are seen due to noise. It is clearly observed that the tracking performance also deteriorated.

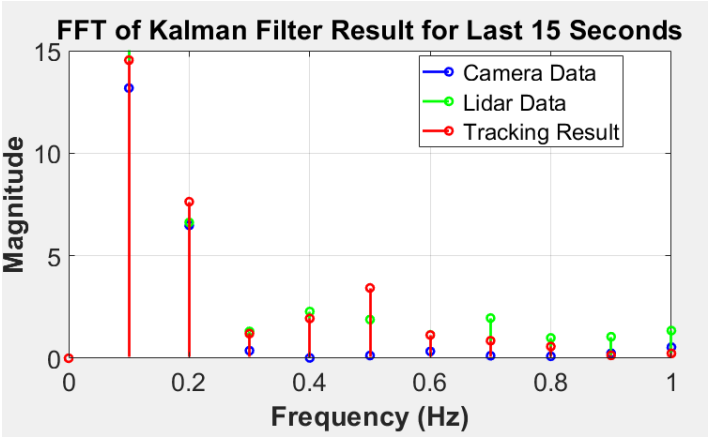

Figure 6.5 Constant Velocity Kalman Filter Model FFT Last 15 Seconds Result with Increase Noise

When the performance of the Kalman filter in Figure 6.6 is examined, it is seen that the estimation performance of the LiDAR data has decreased considerably. As the sensor data are affected by noise and become unreliable, the Kalman filter assumes that the measurement

noise in the LiDAR data is constant. The fact that the calculated noise covariance values are close to each other and the average of the two data also worsens the tracking performance.

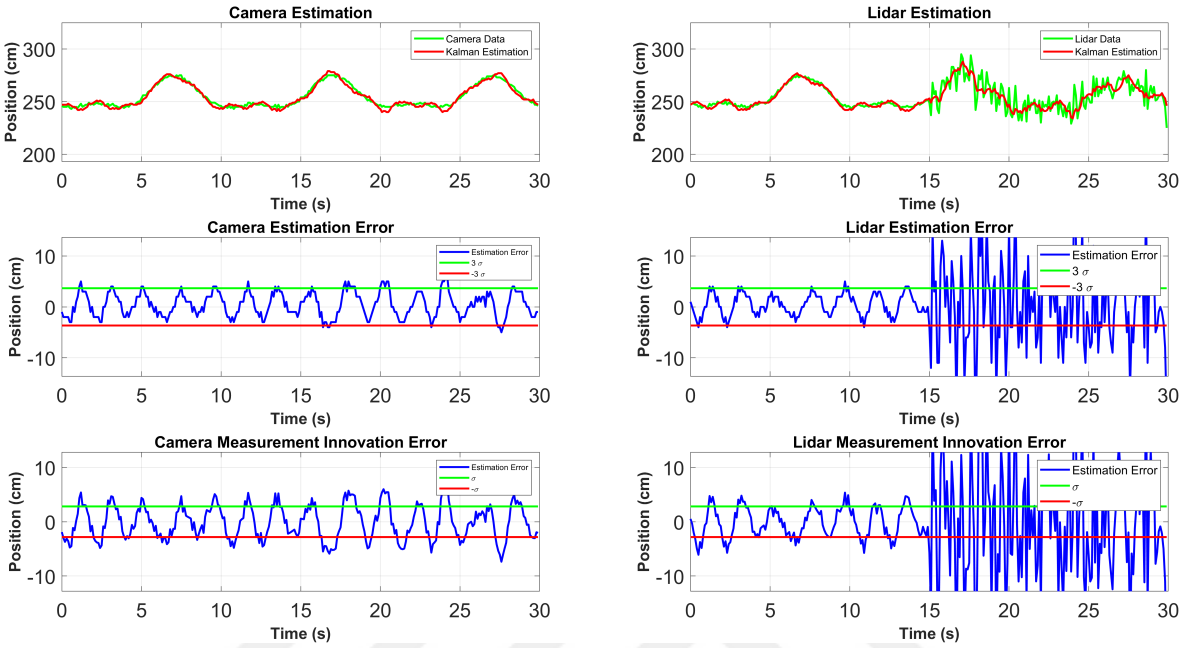

Figure 6.6 Constant Velocity Kalman Filter Model Performance Under Increase Noise

Accurately estimating the measurement noise can help improve the performance of the Kalman filter in such a situation. When we examined the literature, it was seen that the residual-based adaptive estimation method was recommended to estimate measurement noise [43]. In equation (53), the residual is defined as the difference between the measured and estimated value.

$$\varepsilon_k = y_k - Cx_{k+1} \tag{53}$$

When we examine the equation (53), it is seen that the result can be positive or negative. Since measurement noise is expressed as variance, this expression must be positive. For this reason, the equation is multiplied by its transpose. Then the average values in a moving window of length N were calculated. N is equal to 5 in the experiments.

$$\hat{W}_k = \frac{1}{N} \sum_{k-N-1}^{k} \varepsilon_k \varepsilon_k^T \tag{54}$$

The value calculated in the last step is summed with the estimated error covariance value. The proposed equation for measurement noise estimation in the last step is as follows (55).

$$\hat{R}_k = \hat{W}_k + CP_{k+1}C^T \tag{55}$$

Using this equation, the flow of the Kalman filter has been updated as in Algorithm 2 below.

Algorithm 2 KF with Residual Based Measurement Noise Estimation (Moving Window)

Initialize:

$$x_0 = \begin{bmatrix} L_{pos} \\ L_{vel} \\ C_{pos} \\ C_{vel} \end{bmatrix}, P_0 = \begin{bmatrix} P_{L_{pos}} & 0 & 0 & 0 \\ 0 & P_{L_{vel}} & 0 & 0 \\ 0 & 0 & P_{C_{pos}} & 0 \\ 0 & 0 & 0 & P_{C_{vel}} \end{bmatrix}$$

for $k \leftarrow 1$ to N **do**

Prediction:

$x_k = Ax_{k-1}$

$P_k = AP_{k-1}A^T + Q$

Update:

$S = CP_{k-1}C^T + R$

$K = PCS^{-1}$

$x_{k+1} = x_k + K(y_k - x_k)$

$P_{k+1} = (1 - KC)P_k(1 - KC)^T + KRK^T$

$\varepsilon_k = y_k - Cx_{k+1}$

if $k \geq 5$ **then**

$R = \frac{1}{N}\sum_{k-N-1}^{k} \varepsilon_k \varepsilon_k^T + CP_{k+1}C^T$

$R = Diag[R(0,0), R(1,1)]$

end if

Fusion:

$T_{pos} = \frac{P_{C_{pos}} L_{pos}}{P_{L_{pos}} + P_{C_{pos}}} + \frac{P_{L_{pos}} C_{pos}}{P_{L_{pos}} + P_{C_{pos}}}$

end for

The experiment in Figure 6.4 was repeated using this method. When the time domain graph Figure 6.7 is examined, the tracking performance is quite good compared to the previous situation. The RMSE value is low, and according to the result shown in the frequency domain graph, the tracking behavior is very close to the reference signal.

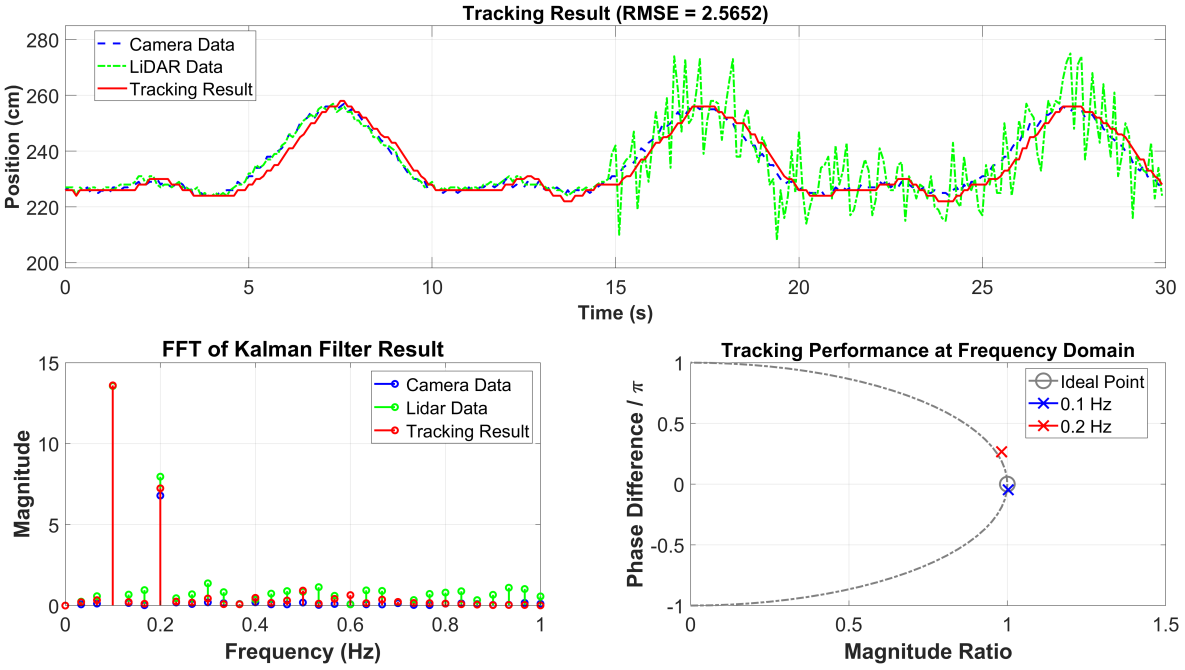

Figure 6.7 Constant Velocity Kalman Filter Model Result with Residual Based Measurement Noise Estimate (Moving Window)

Figure 6.8 shows the performance of the Kalman filter with residual-based measurement noise estimated using the moving window method. The high amount of noise added after 15 seconds corrupted the LiDAR data. After this point, the LiDAR estimation error changed dynamically as the measurement noise was updated. At the same time, the innovation error of the measurement was also updated based on the change. The estimated error value remains in the updated range. Due to the update of the measurement error, the Kalman filter continued to work properly.

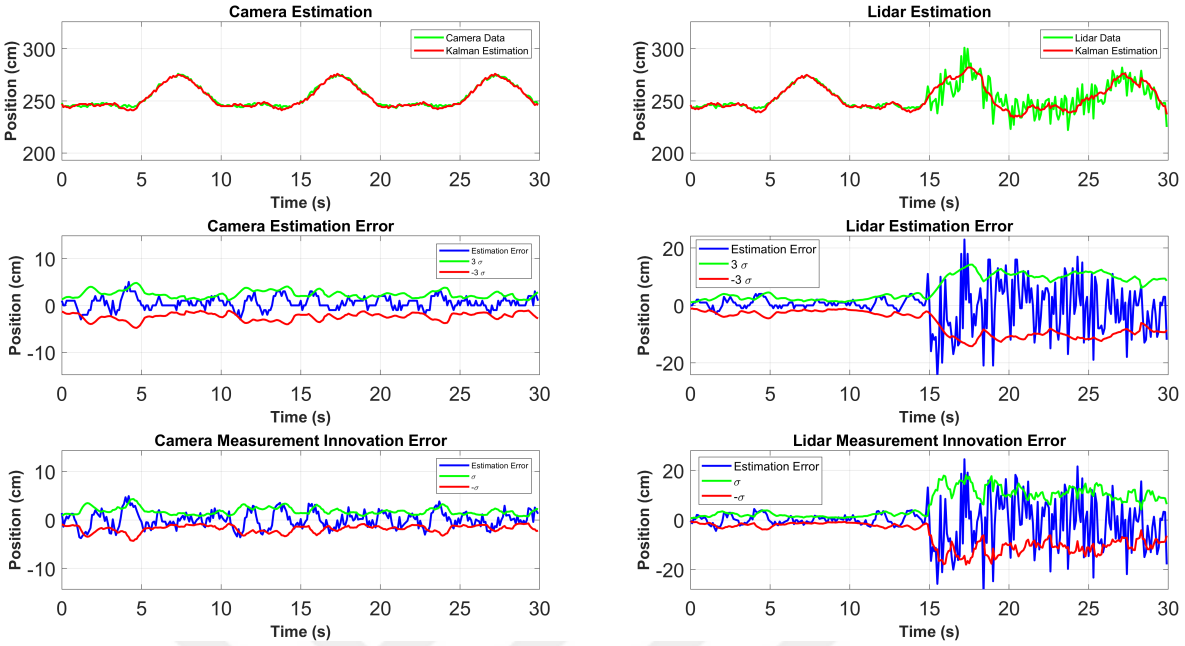

Figure 6.8 Constant Velocity Kalman Filter Model Performance Under Increase Noise with Innovation Based Measurement Noise Estimate (Moving Window)

An increase in the error in LiDAR data means an increase in the error covariance value. This reduces the reliability of data obtained from LiDAR. Since we combine the sensor data with inverse variance weighting in the fusion stage, the weight of the camera data should be high in the results. Figure 6.9 shows the weighting relationship between the camera and LiDAR data. In the first 15 seconds, a zero mean Gaussian noise with the same variance ratio is randomly added to the camera and LiDAR data. Although the error covariance value calculated at this stage is very low, the weights have changed due to randomness. After 15 seconds, since the error in the LiDAR data is higher than the camera data, almost all the weight in the fusion stage is given to the camera data.

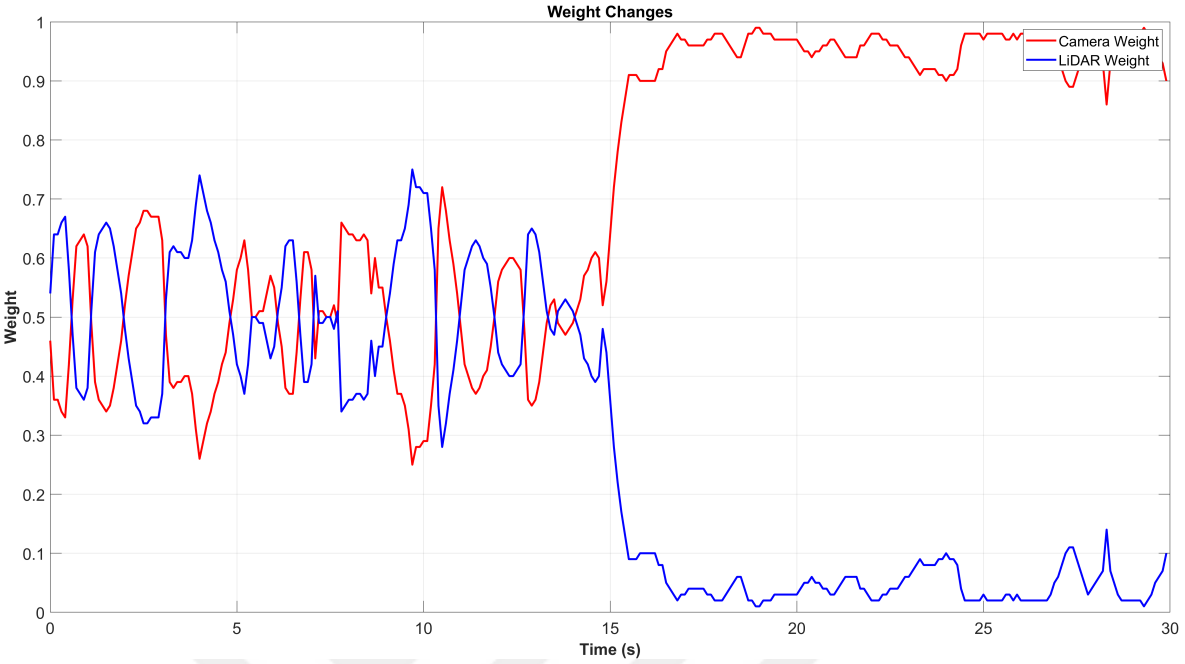

Figure 6.9 Innovation Based Measurement Noise Estimation (Moving Window) Fusion Result Weight Changes

6.4. Kalman Filter Based Sensor Fusion with Residual Based Measurement Noise Estimation (Forgetting Factor)

Instead of using the moving window, the forgetting factor $0 < \alpha < 1$ can be used for estimating to measurement noise [44]. Instead of averaging N number of expressions in a moving window, this method calculates current measurement noise using weighted previous value by alpha. This similar method is expressed in equation (56).

$$\hat{R}_k = \alpha \hat{R}_{k-1} + (1-\alpha)((y - x_{k+1})(y - x_{k+1})^T + CP_{k+1}C^T) \tag{56}$$

Using this equation, the flow of the Kalman filter has been updated as in Algorithm 3 below.

Algorithm 3 KF with Residual Based Measurement Noise Estimation (Forgetting Factor)

Initialize:

$$x_0 = \begin{bmatrix} L_{pos} \\ L_{vel} \\ C_{pos} \\ C_{vel} \end{bmatrix}, P_0 = \begin{bmatrix} P_{L_{pos}} & 0 & 0 & 0 \\ 0 & P_{L_{vel}} & 0 & 0 \\ 0 & 0 & P_{C_{pos}} & 0 \\ 0 & 0 & 0 & P_{C_{vel}} \end{bmatrix}$$

for $k \leftarrow 1$ to N **do**

Prediction:

$$x_k = Ax_{k-1}$$

$$P_k = AP_{k-1}A^T + Q$$

Update:

$$S = CP_{k-1}C^T + R$$

$$K = PCS^{-1}$$

$$x_{k+1} = x_k + K(y_k - x_k)$$

$$P_{k+1} = (1 - KC)P_k(1 - KC)^T + KRK^T$$

$$R = \alpha R + (1 - \alpha)((y - Cx_{k+1})(y - Cx_{k+1})^T + CPC^T)$$

$$R = Diag[R(0,0), R(1,1)]$$

Fusion:

$$T_{pos} = \frac{P_{C_{pos}} L_{pos}}{P_{L_{pos}} + P_{C_{pos}}} + \frac{P_{L_{pos}} C_{pos}}{P_{L_{pos}} + P_{C_{pos}}}$$

end for

This time, the experiment in Figure 6.4 was repeated with forgetting factor-based measurement noise estimation. $\alpha = 0.3$ was chosen. The results are quite similar to the previous method when we examine the figure 6.10 time domain graph and RMSE value. Likewise, the weight relationship between sensors data and the performance of the Kalman filter is quite similar to the previous test results, as shown in figure 6.11 and figure 6.12.

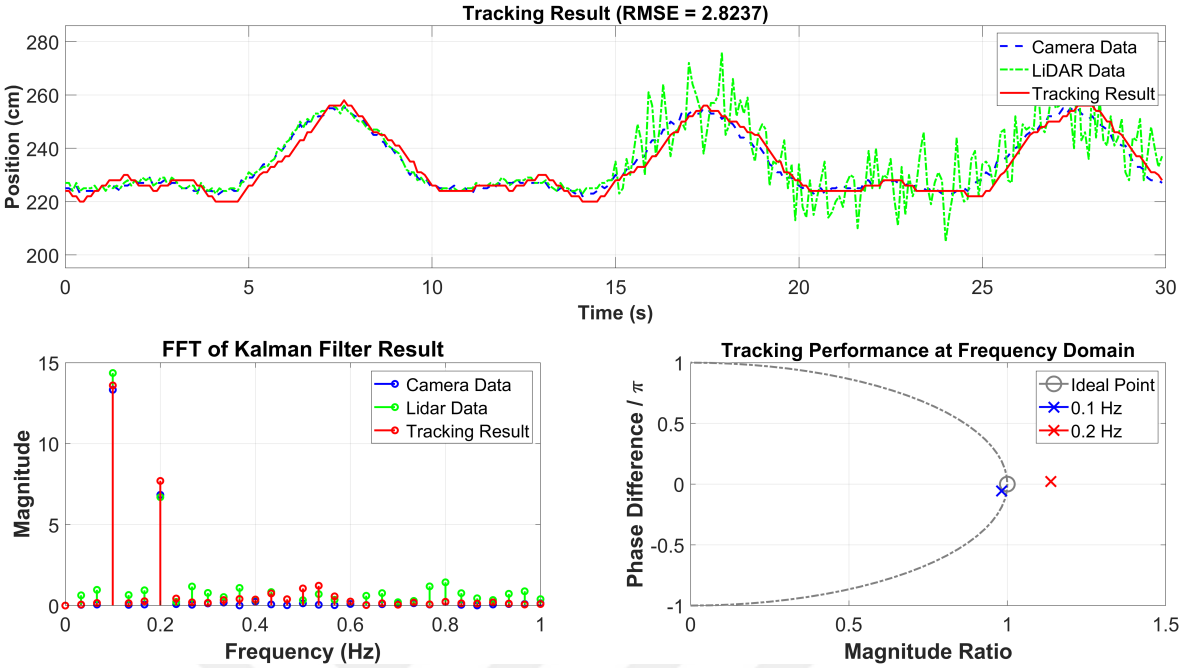

Figure 6.10 Constant Velocity Kalman Filter Model Result with Residual Based Measurement Noise Estimation (Forgetting Factor)

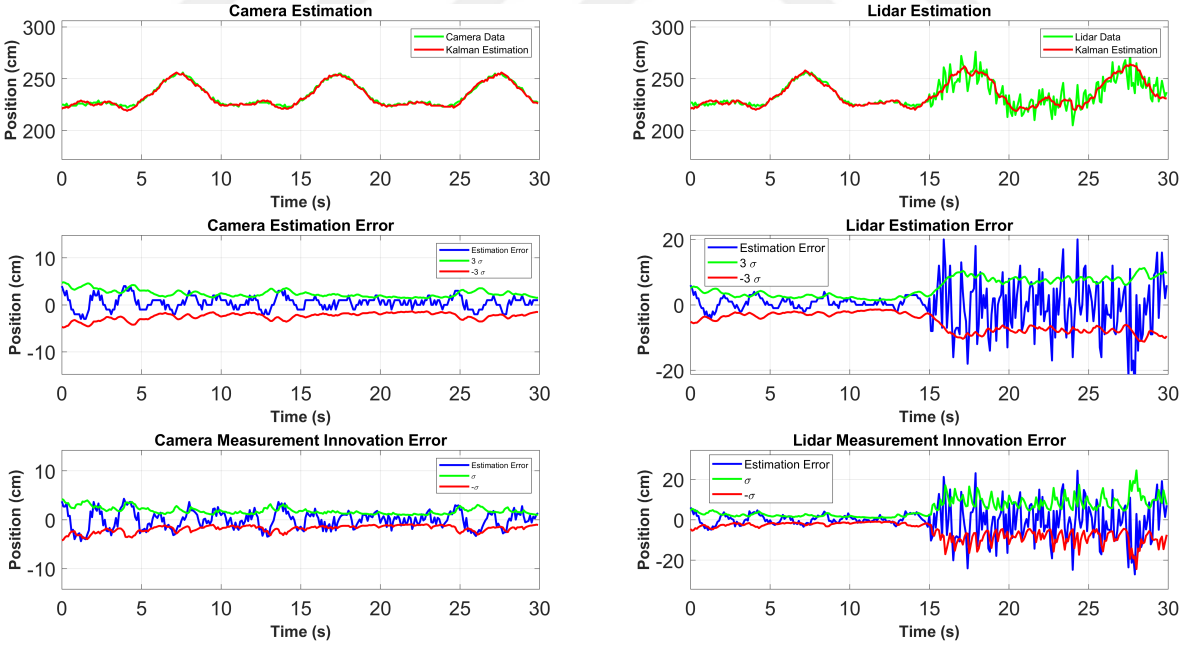

Figure 6.11 Constant Velocity Kalman Filter Model Performance Under Increase Noise with Innovation Based Measurement Noise Estimate (Forgetting Factor)

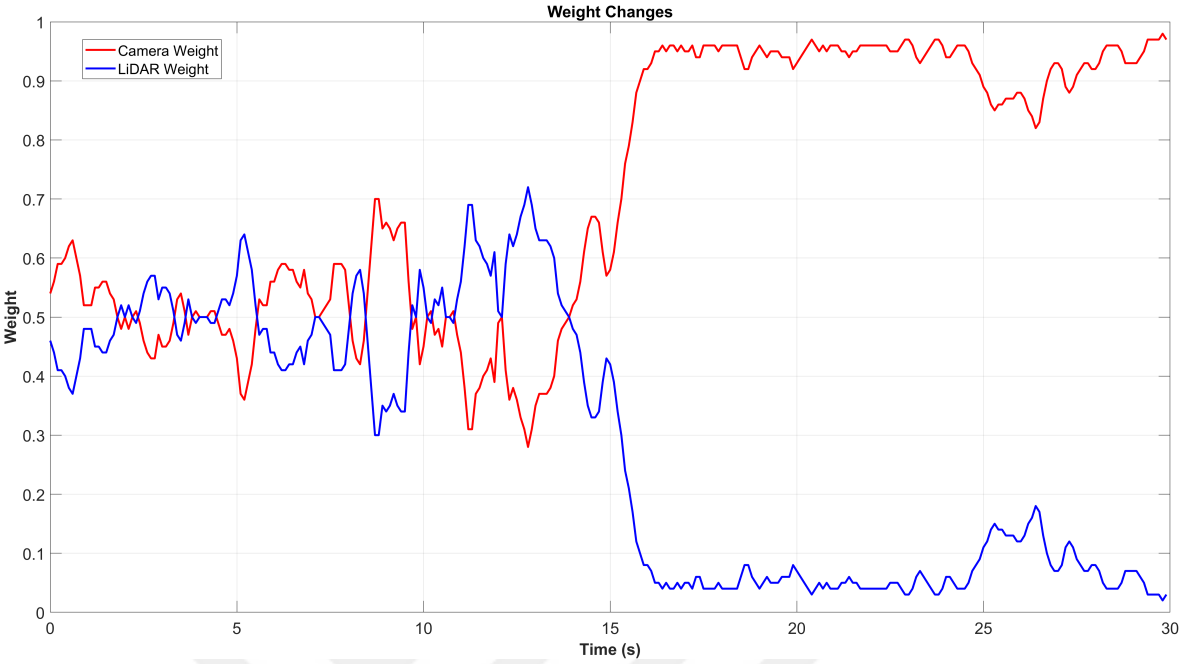

Figure 6.12 Innovation Based Measurement Noise Estimation (Forgetting Factor) Fusion Result Weight Changes

6.5. MCC-KF Based Sensor Fusion

In addition to situations such as the corruption of sensor data by noise and the unreliability of one sensor compared to the other, the sensor data can also be affected by shot noise. While the sensor may cause this shot noise, it may also occur due to an unexpected event in the environment. In this case, it dramatically distorts the measurement result, even if it is instantaneous. In the experiment shown in Figure 6.13, the LiDAR data was randomly corrupted by shot noises. When we examine the time domain results, tracking performance is relatively poor after data is affected by shot noise.

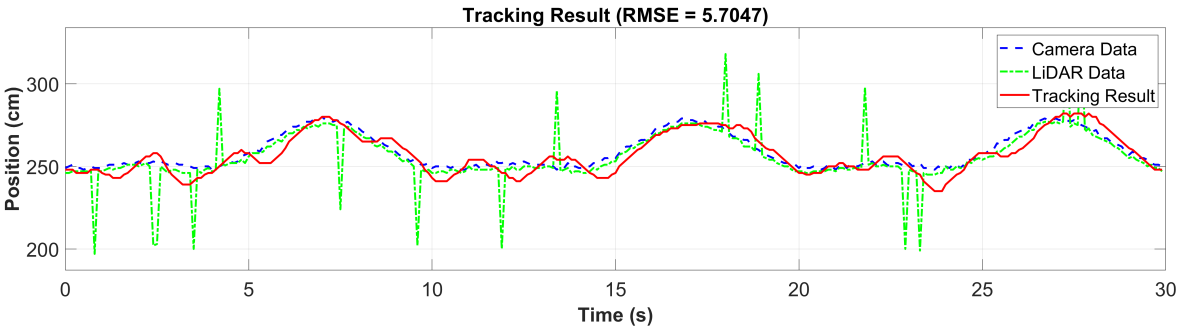

Figure 6.13 Constant Velocity Kalman Filter Model Result with Shot Noise

In addition, when we examine the data in the frequency domain, as shown figure 6.14, the distortions in the LiDAR data, apart from the main frequencies, cause obvious distortions on the tracking data.

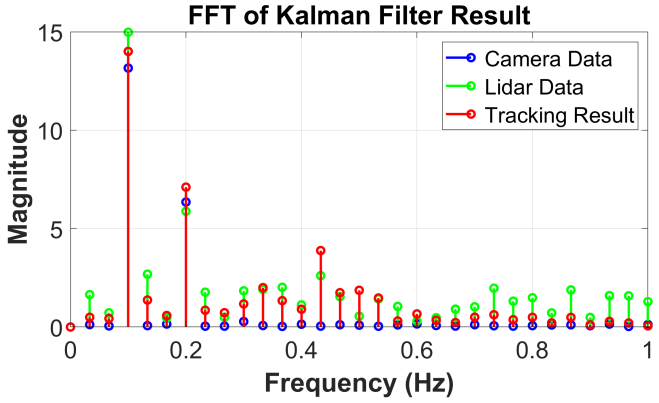

Figure 6.14 Constant Velocity Kalman Filter Model FFT Result with Shot Noise

When we examine the performance of the Kalman filter under shot noise based on all the data, the error covariance and measurement innovation error values estimated by the filter are out of the range. Because in such cases, the traditional Kalman filter cannot show sufficient performance.

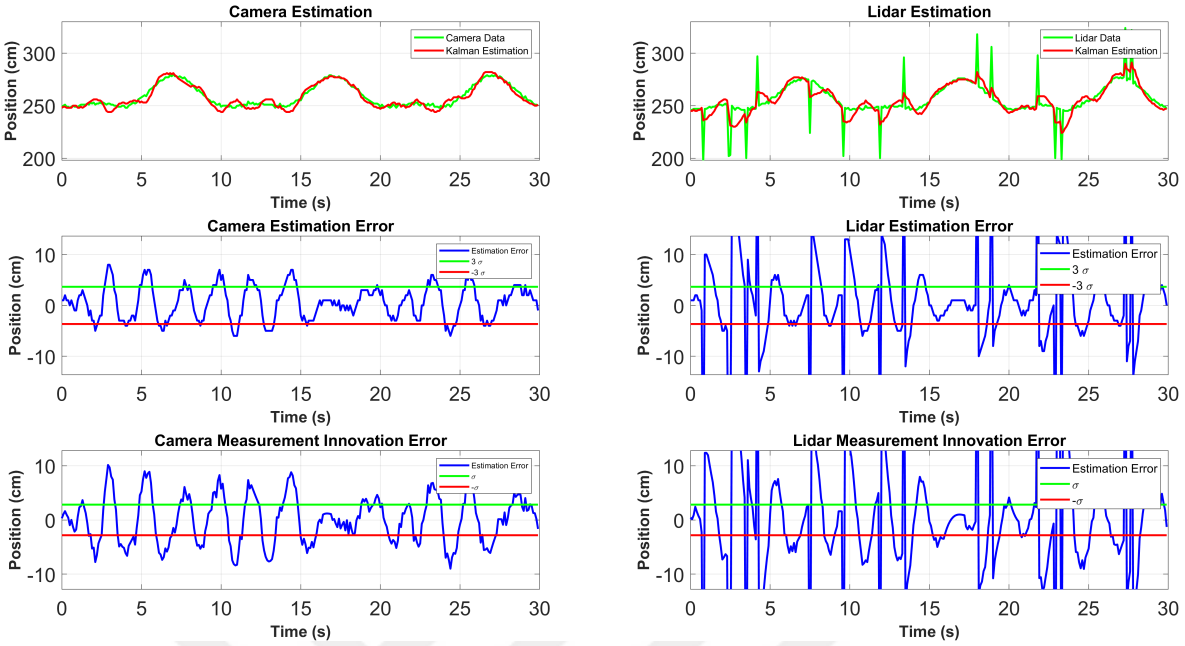

Figure 6.15 Constant Velocity Kalman Filter Model FFT Result with Shot Noise

Several methods to improve the performance of the Kalman filter in cases where the sensor data is affected by both Gaussian noise and shot noise have been proposed in the literature. One of them is the MCC-KF. In this method, the MCC-KF has a structure that weights the Kalman gain value when the noise rises instantaneously. This weighting process is performed using the equation (29) we have defined in the previous sections. G_z is the Gaussian kernel function shown in (25). The z kernel bandwidth size is a key parameter in this equation. If the z is chosen too small compared to the shot noise, MCC-KF very slowly converges to the desired value. On the other hand, if the z is chosen too large, equation converges to 1 and MCC-KF behaves like traditional Kalman filter [45]. In our experiments, a fixed value of $z = 5$ was chosen.

When we choose $z = 5$, the $2z^2$ is quite large compared to the $||\hat{x}_k - A\hat{x}_{k-1}||^2_{P^{-1}_{k|k-1}}$ expression in the denominator of equation (29). In this case, equation (29) can be rearranged as follows equation (57) [46].

$$G_z(||\hat{x}_k - A\hat{x}_{k-1}||_{P^{-1}_{k|k-1}}) \approx 1 \Rightarrow L_k = G_z(||y_k - C\hat{x}_k||_{R_k^{-1}}) \tag{57}$$

In the case we use MCC-KF, we focused on the relationship between the measurement and the estimated value. L_k calculated separately for each sensor data [47] as shown as equation (58).

$$Co = Diag[G_z(||y_L - C_1\hat{x}_k|_{R^{-1}_{(0,0)}}), 0, G_z(||y_C - C_2\hat{x}_k|_{R^{-1}_{(2,2)}}), 0] \tag{58}$$

The Kalman filter flow updated using all this information is shown in Algorithm 4.

Algorithm 4 MCC-KF

Initialize:

$$x_0 = \begin{bmatrix} L_{pos} \\ L_{vel} \\ C_{pos} \\ C_{vel} \end{bmatrix}, P_0 = \begin{bmatrix} P_{L_{pos}} & 0 & 0 & 0 \\ 0 & P_{L_{vel}} & 0 & 0 \\ 0 & 0 & P_{C_{pos}} & 0 \\ 0 & 0 & 0 & P_{C_{vel}} \end{bmatrix}$$

for $k \leftarrow 1$ to N **do**

Prediction:

$x_k = Ax_{k-1}$

$P_k = AP_{k-1}A^T + Q$

Update:

$L_L = (||y_L - C_1\hat{x}_k||_{R^{-1}_{(0,0)}})$

$L_C = (||y_C - C_2\hat{x}_k||_{R^{-1}_{(2,2)}})$

$$Co = \begin{bmatrix} L_L & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & L_C & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$S = CCoP_{k-1}C^T + R$

$K = PCoCS^{-1}$

$x_{k+1} = x_k + K(y_k - x_k)$

$P_{k+1} = (1 - KC)P_k(1 - KC)^T + KRK^T$

Fusion:

$T_{pos} = \frac{P_{C_{pos}}L_{pos}}{P_{L_{pos}}+P_{C_{pos}}} + \frac{P_{L_{pos}}C_{pos}}{P_{L_{pos}}+P_{C_{pos}}}$

end for

In the experiment in Figure 6.13, the performance of the Kalman filter deteriorated considerably when the sensor data was corrupted by instantaneous noise. In the experiment in Figure 6.16, same shot noise case occurred but this time MCC-KF is used. Considering the time domain tracking performance, the MCC-KF gave a very good result in the case of shot noise.

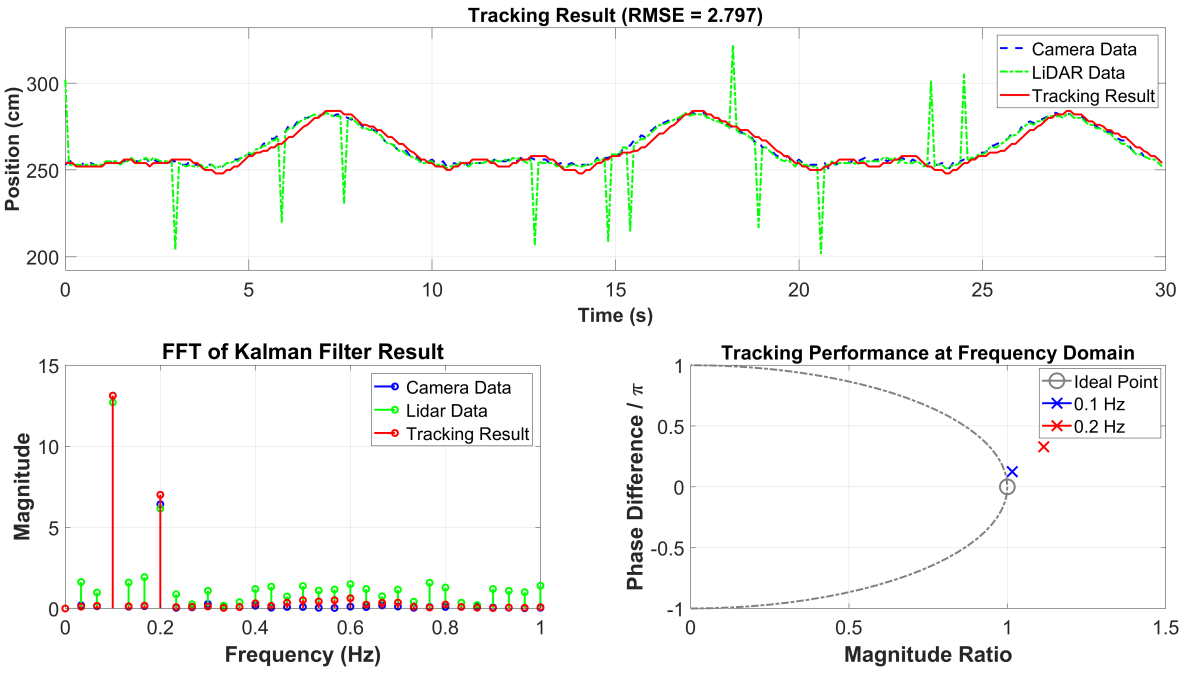

Figure 6.16 Constant Velocity MCC-KF Model Performance Under Shot Noise

When we examine the MCC-KF performance in Figure 6.17 below, we can see that although the difference between the measured value and the estimated value in the LiDAR data increased, the measurement value was neglected by updating the Kalman gain and the tracking performance was improved.

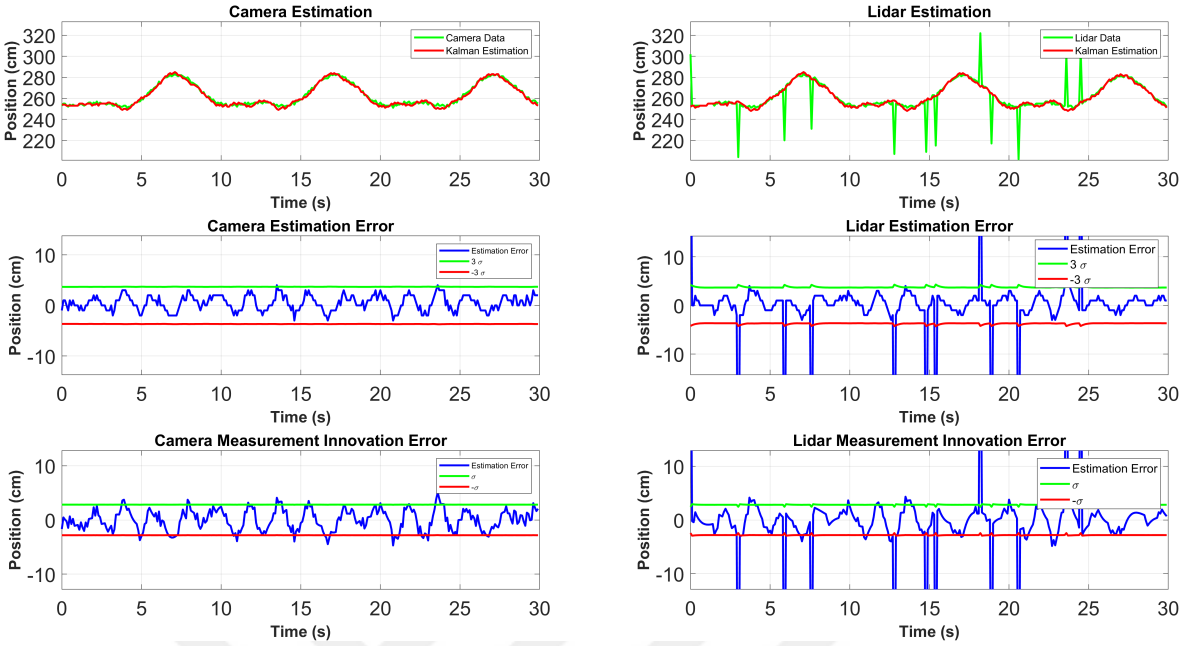

Figure 6.17 Constant Velocity MCC-KF Model Performance Under Shot Noise

Since there is no other noise in the environment while the LiDAR data is corrupted by shot noises, there is a slight change in the LiDAR error covariance value when the noise corrupts the data. In this case, the change of the weight applied to the sensor data every time the shot noise comes is shown in Figure 6.18.

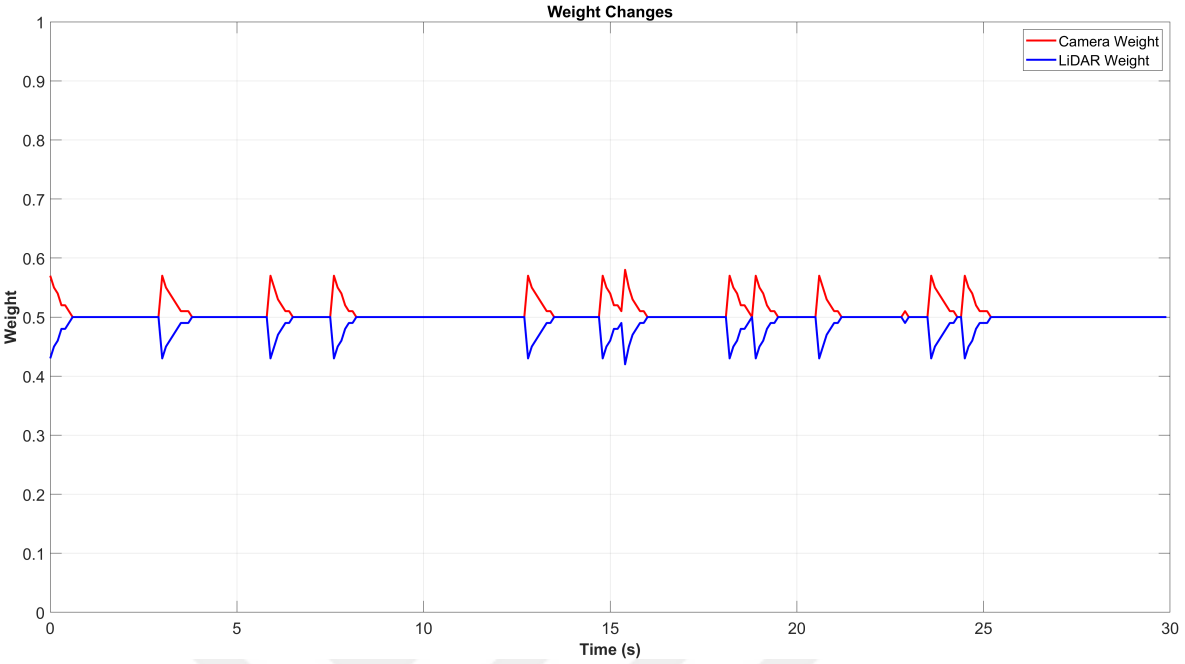

Figure 6.18 MCC-KF Model Fusion Result Weight Changes

6.6. MCC-KF with Measurement Noise Estimation (Moving Window)

Measurement noise estimation methods function by averaging a specific number of computed values or by assigning weight to the previously calculated value. This approach proves particularly effective when the quality of sensor data deteriorates or when noise consistently intensifies within a defined range. On the other hand, MCC can work efficiently in the instantaneous distortions. Below Algorithm 5 is designed for scenarios in which noise increases at specific time intervals and where shot noise is present in the data. In this algorithm, MCC and measurement noise estimation methods are combined. MCC is used first and then measurement noise estimation is used.

Algorithm 5 MCC-KF with Measurement Noise Estimation (Moving Window)

Initialize:

$$x_0 = \begin{bmatrix} L_{pos} \\ L_{vel} \\ C_{pos} \\ C_{vel} \end{bmatrix}, P_0 = \begin{bmatrix} P_{L_{pos}} & 0 & 0 & 0 \\ 0 & P_{L_{vel}} & 0 & 0 \\ 0 & 0 & P_{C_{pos}} & 0 \\ 0 & 0 & 0 & P_{C_{vel}} \end{bmatrix}$$

for $k \leftarrow 1$ to N **do**

Prediction:

$x_k = Ax_{k-1}$

$P_k = AP_{k-1}A^T + Q$

Update:

$L_L = (||y_L - C_1\hat{x}_k||_{R^{-1}_{(0,0)}})$

$L_C = (||y_C - C_2\hat{x}_k||_{R^{-1}_{(2,2)}})$

$$Co = \begin{bmatrix} L_L & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & L_C & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$S = CCoP_{k-1}C^T + R$

$K = PCoCS^{-1}$

$x_{k+1} = x_k + K(y_k - x_k)$

$P_{k+1} = (1 - KC)P_k(1 - KC)^T + KRK^T$

$\varepsilon_k = y_k - Cx_{k+1}$

if $k \geq 5$ **then**

$R = \frac{1}{T}\sum_{k-T-1}^{k} \varepsilon_k\varepsilon_k^T + CP_{k+1}C^T$

$R = Diag[R(0,0), R(1,1)]$

end if

Fusion:

$T_{pos} = \frac{P_{C_{pos}}L_{pos}}{P_{L_{pos}}+P_{C_{pos}}} + \frac{P_{L_{pos}}C_{pos}}{P_{L_{pos}}+P_{C_{pos}}}$

end for

To test the algorithm, the LiDAR data was corrupted by both shot noise and a Gaussian noise after a certain time. Upon reviewing Figure 6.19, while LiDAR data is corrupted by random instantaneous noise, in addition, Gaussian noise with a different amplitude was added to the data after the 15th second. As seen in the time domain result, a low RMSE value was obtained as a result of the experiment. The system's tracking performance exhibited a delay in the high-frequency component. Nevertheless, it can be affirmed that the tracking performance remains quite good despite the data corruption within the LiDAR.

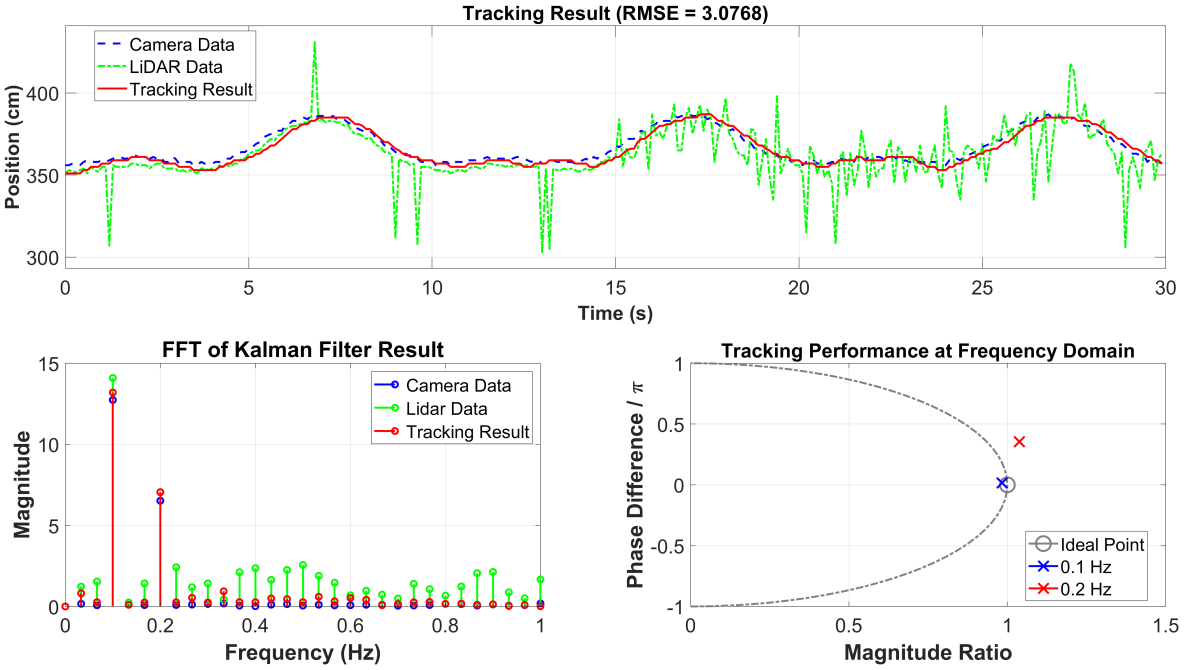

Figure 6.19 MCC-KF with Measurement Noise Estimation (Moving Window) Model Result

Although the noise increased after 15 seconds in the result of the LiDAR sensor in Figure 6.20 below, the estimated noise did not increase that much. Because it is due to the limitation of the σ value in the MCC equation. The update enabled by MCC in the gain allowed us to achieve an efficient outcome, even though the noise estimation was not computed at the desired levels.

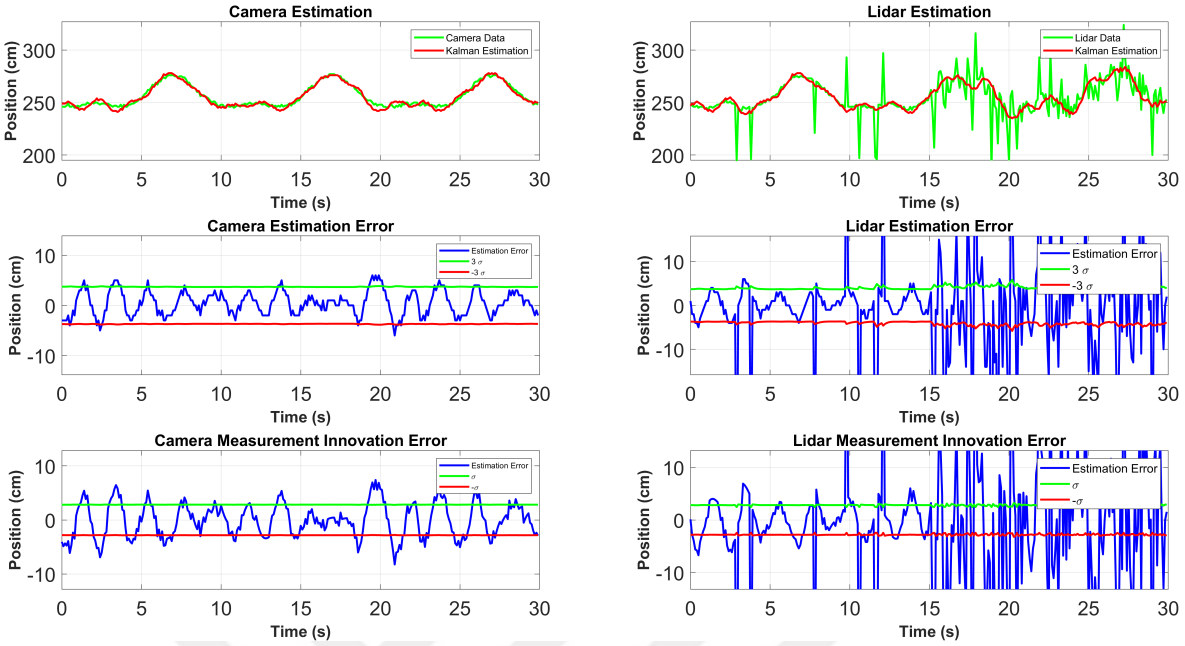

Figure 6.20 MCC-KF with Measurement Noise Estimation (Moving Window) Model Performance Under Shot and Increase Noise

If we had performed measurement noise estimation using a moving window or forgetting factor during an instance of shot noise, we would have obtained misleading results since the subsequent 4 data points would yield high measurement noise estimates. On the other hand, if we used MCC in a situation where the noise was increased, the tracking performance would converge to an incorrect value, as we would always tend to rely on the previous value. Using both methods allows us to avoid this situation. Even though there was a loss in measurement noise estimation performance due to MCC while the noise increased, the weighting could be kept at the desired level as seen in Figure 6.21.

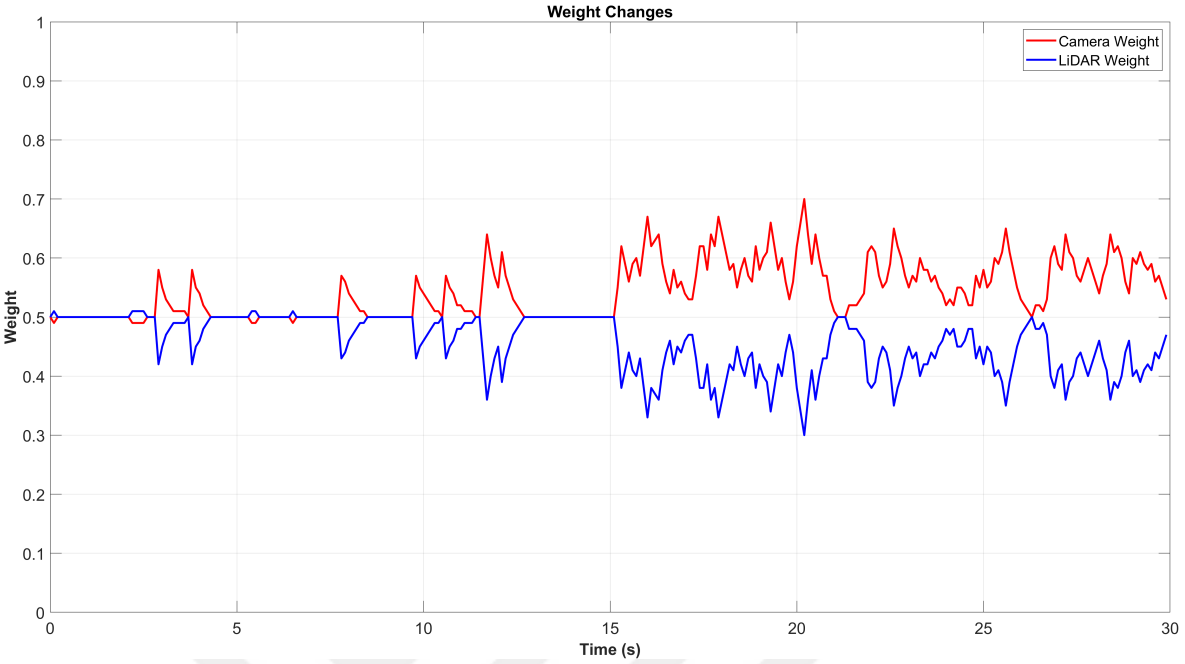

Figure 6.21 MCC-KF with Measurement Noise Estimation (Moving Window) Model Fusion Result Weight Changes

7. CONCLUSION

In this thesis, we prepared a simulation environment to test sensor fusion algorithms. For this purpose, we used an open-source robot with different sensors. This simulation environment, which was created independently of the hardware, proved to be very useful for quickly testing algorithms and identifying potential errors that could be encountered at the hardware level. Additionally, the software developed in this environment can also be used for various sensor fusion studies.

As a result of the studies in the simulation environment, an experimental setup was prepared to conduct experiments on a physical system. Thanks to this setup, the results obtained in the simulation environment were validated on a real robotic system. The results obtained here provide a foundation for engineering systems research.

A motion set was defined for the target objects on this experimental setup, and the mechanism on the sensors initially attempted to track this motion using a superposition-based fusion method. Subsequently, system noise was injected, and since the tracking performance deteriorated, efforts were made to improve the system performance by using the constant velocity Kalman filter model.

The Kalman filter provides optimal results under certain conditions. However, its effectiveness decreases when sensor data deteriorates over time and when data is affected by shot noise. To address this issue, methods from the literature were examined. The measurement noise value was not assumed to be constant, and methods were attempted to estimate it based on the difference between estimation and measurement. In cases of increased noise, this method successfully updated the error covariance value obtained from the Kalman filter's results. Additionally, the MCC-KF was used, especially to enhance the estimation performance of signals affected by shot noise. Subsequently, both methods' advantages were combined to fuse sensor data. As a result, dynamic reweighting was applied to improve tracking performance.

In future studies, sensor fusion methods such as the Extended Kalman Filter and Unscented Kalman Filter can be used in systems that involve different sensor models. Performance comparisons can be made with the results obtained here. The value of z, which holds critical importance particularly for MCC-KF, can be adaptively selected in parallel with the noise in the system to enhance the filter's performance under different noise conditions. Additionally, new studies can be conducted to develop a combined algorithm that applies both MCC-KF and measurement noise estimation methods.

REFERENCES

[1] Muhammed Seyda Comertler and Ismail Uyanik. Salience of multisensory feedback regulates behavioral variability. *Bioinspiration & Biomimetics*, 17(1):016006, **2021**.

[2] Ismail Uyanik, Sarah A Stamper, Noah J Cowan, and Eric S Fortune. Sensory cues modulate smooth pursuit and active sensing movements. *Frontiers in behavioral neuroscience*, 13:59, **2019**.

[3] Ismail Uyanik, Shahin Sefati, Sarah A Stamper, Kyoung-A Cho, M Mert Ankarali, Eric S Fortune, and Noah J Cowan. Variability in locomotor dynamics reveals the critical role of feedback in task control. *Elife*, 9:e51219, **2020**.

[4] D.L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, **1997**.

[5] R.C. Luo and M.G. Kay. Multisensor integration and fusion in intelligent systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):901–931, **1989**.

[6] R.C. Luo and M.G. Kay. A tutorial on multisensor integration and fusion. In *[Proceedings] IECON '90: 16th Annual Conference of IEEE Industrial Electronics Society*, pages 707–722 vol.1. **1990**.

[7] Jan Drugowitsch, Gregory C DeAngelis, Eliana M Klier, Dora E Angelaki, and Alexandre Pouget. Optimal multisensory decision-making in a reaction-time task. *eLife*, 3:e03005, **2014**.

[8] Hirokawa J Sadakane O Sakata S Bosch M Sakurai Y Yamamori T. Multisensory information facilitates reaction speed by enlarging activity difference between superior colliculus hemispheres in rats. *PLoS One*, 6(9):e25283, **2011**.

[9] Mark A. Frye and Michael H. Dickinson. Motor output reflects the linear superposition of visual and olfactory inputs in Drosophila. *Journal of Experimental Biology*, 207(1):123–131, **2004**.

[10] Erin E. Sutton, Alican Demir, Sarah A. Stamper, Eric S. Fortune, and Noah J. Cowan. Dynamic modulation of visual and electrosensory gains for locomotor control. *Journal of The Royal Society Interface*, 13(118), **2016**.

[11] Shane Windsor, Richard Bomphrey, and Graham Taylor. Vision-based flight control in the hawkmoth hyles lineata. *Journal of the Royal Society, Interface / the Royal Society*, 11:20130921, **2014**.

[12] Noah Cowan, Mustafa Ankarali, Jonathan Dyhr, Manu Madhav, Eatai Roth, Shahin Sefati, Simon Sponberg, Sarah Stamper, Eric Fortune, and Thomas Daniel. Feedback control as a framework for understanding tradeoffs in biology. *Integrative and comparative biology*, 54, **2014**.

[13] Jelena Kocić, Nenad Jovičić, and Vujo Drndarević. Sensors and sensor fusion in autonomous vehicles. In *2018 26th Telecommunications Forum (TELFOR)*, pages 420–425. **2018**.

[14] Dhananjay Singh, Gaurav Tripathi, and Antonio J. Jara. A survey of internet-of-things: Future vision, architecture, challenges and services. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 287–292. **2014**.

[15] Alper Yilmaz. Sensor fusion in computer vision. In *2007 Urban Remote Sensing Joint Event*, pages 1–5. **2007**.

[16] Man Lok Fung, Michael Z. Q. Chen, and Yong Hua Chen. Sensor fusion: A review of methods and applications. In *2017 29th Chinese Control And Decision Conference (CCDC)*, pages 3853–3860. **2017**.

[17] R.C. Luo, Chih-Chen Yih, and Kuo Lan Su. Multisensor fusion and integration: approaches, applications, and future research directions. *IEEE Sensors Journal*, 2(2):107–119, **2002**.

[18] D. Smith and S. Singh. Approaches to multisensor data fusion in target tracking: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 18(12):1696–1710, **2006**.

[19] Ramakrishna Gurajala, Praveen B. Choppala, James Stephen Meka, and Paul D. Teal. Derivation of the kalman filter in a bayesian filtering perspective. In *2021 2nd International Conference on Range Technology (ICORT)*, pages 1–5. **2021**.

[20] Nikulin M.S. Voinov V. G. *Unbiased Estimators and Their Applications Volume 1: Univariate Case*. Kluwer Academic Publishers, **1993**.

[21] Arthur Gelb and The Analytic Sciences Corporation. *Applied Optimal Estimation*. The MIT Press, **1974**.

[22] Franklin A. Graybill and R. B. Deal. Combining unbiased estimators. *Biometrics*, 15(4):543–550, **1959**.

[23] Xuyou Li, Yanda Guo, and Qingwen Meng. Variational bayesian-based improved maximum mixture correntropy kalman filter for non-gaussian noise. *Entropy*, 24(1), **2022**.

[24] Xi Liu, Hua Qu, Jihong Zhao, and Badong Chen. Extended kalman filter under maximum correntropy criterion. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1733–1737. **2016**.

[25] Weifeng Liu, Puskal P. Pokharel, and Jose C. Principe. Correntropy: Properties and applications in non-gaussian signal processing. *IEEE Transactions on Signal Processing*, 55(11):5286–5298, **2007**.

[26] Seyed Fakoorian, Alireza Mohammadi, Vahid Azimi, and Dan Simon. Robust kalman-type filter for non-gaussian noise: Performance analysis with unknown noise covariances. *Journal of Dynamic Systems, Measurement, and Control*, 141, **2019**.

[27] Reza Izanloo, Seyed Abolfazl Fakoorian, Hadi Sadoghi Yazdi, and Dan Simon. Kalman filtering based on the maximum correntropy criterion in the presence of non-gaussian noise. In *2016 Annual Conference on Information Science and Systems (CISS)*, pages 500–505. **2016**.

[28] Orhun Koç, Alp Demirel, Emin Yusuf Aydın, Fatmagül İbişoğlu, Şevval İzel Solmaz, Kaan Arı, Ayşe İdman, and İsmail Uyanık. System identification of the target tracking behavior of zebrafish during rheotaxis. In *2022 30th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. **2022**.

[29] Pedro Medeiros de Assis Brasil, Fabio Ugalde Pereira, Marco Antonio de Souza Leite Cuadros, Anselmo Rafael Cukla, and Daniel Fernando Tello Gamarra. A study on global path planners algorithms for the simulated turtlebot 3 robot in ros. In *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, pages 1–6. **2020**.

[30] Wei Qian, Zeyang Xia, Jing Xiong, Yangzhou Gan, Yangchao Guo, Shaokui Weng, Hao Deng, Ying Hu, and Jianwei Zhang. Manipulation task simulation using ros and gazebo. In *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, pages 2594–2598. **2014**.

[31] Daniel Fernando Tello Gamarra, Andrei Piccinini Legg, Marco Antonio de Souza Leite Cuadros, and Emanueli Santos da Silva. Sensory integration of a mobile robot using the embedded system odroid-xu4 and ros. In *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, pages 198–203. **2019**.

[32] Ian Peake, Joseph La Delfa, Ronal Bejarano, and Jan Olaf Blech. Simulation components in gazebo. In *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, volume 1, pages 1169–1175. **2021**.

[33] Farzan M. Noori, David Portugal, Rui P. Rocha, and Micael S. Couceiro. On 3d simulators for multi-robot systems in ros: Morse or gazebo? In *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pages 19–24. **2017**.

[34] Lars Kunze, Tobias Roehm, and Michael Beetz. Towards semantic robot description languages. In *2011 IEEE International Conference on Robotics and Automation*, pages 5589–5595. **2011**.

[35] Denis Chikurtev. Mobile robot simulation and navigation in ros and gazebo. In *2020 International Conference Automatics and Informatics (ICAI)*, pages 1–6. **2020**.

[36] Junior C. Jesus, Jair A. Bottega, Marco A. S. L. Cuadros, and Daniel F. T. Gamarra. Deep deterministic policy gradient for navigation of mobile robots in simulated environments. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 362–367. **2019**.

[37] Alexandru-Calin Stan. A decentralised control method for unknown environment exploration using turtlebot 3 multi-robot system. In *2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6. **2022**.

[38] Ariel Y. Ramos Ruiz and Balasubramaniyan Chandrasekaran. A robotic control system using robot operating system and matlab for sensor fusion and human-robot interaction. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0550–0555. **2020**.

[39] Eatai Roth, Robert Hall, Thomas Daniel, and Simon Sponberg. Integration of parallel mechanosensory and visual pathways resolved through sensory conflict. *Proceedings of the National Academy of Sciences of the United States of America*, 113, **2016**.

[40] E Roth, S Sponberg, and NJ Cowan. A comparative approach to closed-loop computation. *Current Opinion in Neurobiology*, 25:54–62, **2014**.

[41] H. B. Mitchell. *Multi-sensor data fusion an introduction*. Springer Verlag, **2007**.

[42] Alex Becker. *Kalman Filter from the Ground Up*. Alex Becker, **2023**.

[43] K. P. Schwarz A. H. Mohamed. Adaptive kalman filtering for ins/gps. *Journal of Geodesy*, 73, **1999**.

[44] Shahrokh Akhlaghi, Ning Zhou, and Zhenyu Huang. Adaptive adjustment of noise covariance in kalman filter for dynamic state estimation. In *2017 IEEE Power Energy Society General Meeting*, pages 1–5. **2017**.

[45] Badong Chen, Xi Liu, Haiquan Zhao, and José C Príncipe. Maximum correntropy kalman filter. *Automatica*, 76:70–77, **2017**.

[46] Goktug T. Cinar and José C. Príncipe. Hidden state estimation using the correntropy filter with fixed point update and adaptive kernel size. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. **2012**.

[47] Seyed Fakoorian, Reza Izanloo, Azin Shamshirgaran, and Dan Simon. Maximum correntropy criterion kalman filter with adaptive kernel size. In *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, pages 581–584. **2019**.