REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

# IMAGE PROCESSING BASED NAVIGATION IN AVIATION

**Hasan GENCO**

MASTER OF SCIENCE THESIS
Department of Avionics Engineering
Program of Avionics Engineering (Interdisciplinary)

Supervisor
Assoc. Prof. Dr. Ufuk SAKARYA

Co-supervisor
Prof.Dr. Vasfi Emre ÖMÜRLÜ

February, 2023

**REPUBLIC OF TURKEY**
**YILDIZ TECHNICAL UNIVERSITY**
**GRADUATE SCHOOL OF SCIENCE AND ENGINEERING**

# IMAGE PROCESSING BASED NAVIGATION IN AVIATION

A thesis submitted by Hasan GENCO in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 09.02.2023 in Department of Avionics Engineering, Program of Avionics Engineering (Interdisciplinary).

Assoc. Prof. Dr. Ufuk SAKARYA
Yildiz Technical University
Supervisor

Prof.Dr. Vasfi Emre ÖMÜRLÜ
Yildiz Technical University
Co-supervisor

**Approved By the Examining Committee**

Assoc. Prof. Dr. Ufuk SAKARYA, Supervisor
Yildiz Technical University ______________________

Assoc. Prof. Emre ÖZER , Member
İstanbul University-Cerrahpaşa ______________________

Assoc. Prof. Peyman MAHOUTİ, Member
Yildiz Technical University ______________________

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of IMAGE PROCESSING BASED NAVIGATION IN AVIATION supervised by my supervisor, Assoc. Prof. Dr. Ufuk SAKARYA. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Hasan GENCO

Signature

*Dedicated to my family*
*and all the people who supported, inspired, and wished me success throughout my study*

# ACKNOWLEDGEMENTS

I want to acknowledge and express my sincere gratitude to my thesis advisor the head of the Aviation Electrics and Electronics department at Yildiz Technical university Prof. Dr. Ufuk SAKARYA and my thesis Co-Advisor Prof. Dr. Vasfi Emre ÖMÜRLÜ for their continuous support, which made this work possible. Their guidance and advice carried me through all the stages of researching and writing my project.

Besides my thesis advisors, I would like to thank the rest of my thesis committee: Assoc. Prof. Emre ÖZER and Assoc. Prof. Peyman MAHOUTİ for their encouragement, insightful comments, and suggestions.

I would like to express my thanks to Said Burak GÜZEL and Berkay GÜRKAN for their contribution to designing the UNITY engine-based 3D environment which made a significant effect in facilitating training and test data acquisition in this research.

Last but not least, I would love to thank my beloved family who has always stood beside me, supporting me with all my decisions throughout my study.

Hasan GENCO

# TABLE OF CONTENTS

# LIST OF SYMBOLS

| | |
|---|---|
| $\hat{x}$ | An estimate of a state variable X |
| $\theta$ | Angle of rotation |
| Hz | Hertz |
| m | Meters |
| $X^-$ | Predicted (prior) estimate of state variable X |
| s | Seconds |
| $\tilde{y}$ | The Measurement residual |
| $X^+$ | Updated (posterior) estimate of state variable X |

# LIST OF ABBREVIATIONS

| | |
|---|---|
| DoG | Difference of Gaussian |
| GNSS | Global navigation satellite system |
| IMU | Inertial measurement unit |
| INS | Inertial navigation system |
| KF | Kalman Filter |
| LoG | Laplacian of Gaussian |
| RMSE | Root means square error |
| SIFT | Scale-invariant feature transform |
| SURF | Speeded up robust features |
| 1Kmeas KF | One Gain Kalman filter sensor fusion method |
| 3K KF | Three Gains Kalman filter sensor fusion method |

# LIST OF FIGURES

# LIST OF TABLES

# ABSTRACT

## IMAGE PROCESSING BASED NAVIGATION IN AVIATION

Hasan GENCO

Department of Avionics Engineering
Master of Science Thesis

Supervisor: Assoc. Prof. Dr. Ufuk SAKARYA
Co-supervisor: Prof.Dr. Vasfi Emre ÖMÜRLÜ

The importance of navigation safety for autonomous unmanned aerial vehicles (UAVs) has been drastically increasing recently. The increment of this need in the navigation domain has always been associated with rapid development in information technologies and meeting the customer's increasing demand. In this thesis, we present a vision-based multi-sensor fusion navigation method that is robust to obtain precise positioning information despite the loss of one of the measurement signals or having instance bad measurements. The proposed approach included utilizing accelerometer sensors, Global Navigation Satellite System (GNSS) based sensors, and image processing-based sensor systems for navigation in UAVs. Four combinations of scale and rotation invariant interest-point selection-based image processing approaches are used in the proposed method. The selected interest points and the ground truth landmark locations are all together analyzed; then, the position estimation using image processing is obtained. Image processing-based position values are fused with the accelerometer sensor and the GNSS-based sensor values. Two Kalman Filter-based multi-sensor fusion approaches are used in the fusion step of the proposed method. Experiments were demonstrated using a Unity Engine-based generated 3d environment. Different landmarks have been created with various visible appearances and locations, and have been used to simulate UAV flights and record simulation data. The results of the experiments included comparing the proposed method (the fusion of the image processing-based position estimation with the accelerometer sensor and the GNSS-based sensor values) with the baseline method (the fusion of the accelerometer sensor and the GNSS-based sensor values). The results show high accuracy in the UAV positioning estimation algorithm and reliability of the fusion algorithm despite the existence of some significant flight maneuvers and imperfect measurement signals. Which reveals a high-performance promising method suitable for navigation applications in UAVs.

**Keywords:** Unmanned aerial vehicle, image processing, RANSAC, adaptive Kalman filter, sensor fusion

**YILDIZ TECHNICAL UNIVERSITY**
**GRADUATE SCHOOL OF SCIENCE AND ENGINEERING**

# ÖZET

## HAVACILIKTA GÖRÜNTÜ İŞLEME TEMELLİ SEYRÜSEFER

Hasan GENCO

Aviyonik Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Doç. Dr. Ufuk SAKARYA
Eş-Danışman: Prof.Dr. Vasfi Emre ÖMÜRLÜ

Otonom insansız hava araçları (İHA'lar) için seyrüsefer güvenliğinin önemi son zamanlarda büyük ölçüde artmaktadır. Navigasyon alanındaki bu ihtiyacın artması, her zaman bilgi teknolojilerindeki hızlı gelişme ve müşterinin artan talebinin karşılanması ile ilişkilendirilmiştir. Bu tezde, navigasyon sisteminde ölçüm sinyallerinden birinin kaybına veya hatalı ölçümlere sahip olmasına rağmen kesin konumlandırma bilgisi elde etmek için sağlam bir görüntü tabanlı çok sensörlü füzyon navigasyon metodu sunulmaktadır. Önerilen yaklaşım, insansiz hava araçların navigasyon sistemi için ivmeölçer sensörleri, Küresel Navigasyon Uydu Sistemi (GNSS) tabanlı sensörler ve görüntü işleme tabanlı sensör sistemlerini kullanmayı içermektedir. Önerilen yöntemde ölçek ve döndürme değişmez ilgi noktası seçimine dayalı görüntü işleme yaklaşımlarının dört kombinasyonu kullanılmıştır. Seçilen ilgi noktaları ve yer gerçeği yer kontrol noktası konumları hep birlikte analiz edilip; ondan sonra, görüntü işleme kullanarak konum tahmini elde edilmektedir. Görüntü işleme tabanlı konum değerleri, ivmeölçer sensörü ve GNSS tabanlı sensör değerleri ile birleştirilmiştir. Önerilen yöntemin füzyon adımında iki adet Kalman Filtresi tabanlı çok sensörlü füzyon yaklaşımı kullanılmıştır. Deneyler, Unity Engine tabanlı oluşturulmuş bir 3 boyutlu ortam kullanılarak kanıtlanmıştır. İnsansiz hava aracın uçuşlarını simüle etmek ve simülasyon verilerini kaydetmek için farklı yer işaretleri oluşturulmuştur. Yer işaretleri çeşitli görünümlere ve konumlara sahiptir. Deneylerin sonuçları, önerilen yöntemin (görüntü işleme tabanlı konum tahmininin ivmeölçer sensörü ve GNSS tabanlı sensör değerleri ile birleştirilmesi) karşılaştırılan yöntem (ivmeölçer sensörünün GNSS tabanlı sensör değerleri ile birleştirilmesi) ile karşılaştırılmasını içermektedir. Sonuçlar, bazı büyük uçuş manevraları ve hatalı ölçüm sinyallerinin varlığına rağmen insansiz hava aracın konumlandırma tahmin algoritmasında yüksek doğruluk ve füzyon algoritmasının güvenilirliğini göstermektedir. Elde edilen sonuçlar insansiz hava araçlarında navigasyon uygulamalarına uygun, yüksek performanslı, umut verici bir yöntem ortaya çıkarmaktadır.

**Anahtar Kelimeler:** İnsansiz Hava Aracı, görüntü işleme, RANSAC, adaptif Kalman filtresi, sensör füzyonu

# 1
# INTRODUCTION

## 1.1 Literature Review

One of the most essential parts of aerial navigation missions is the challenge of having a trustworthy navigation system. A system that is guaranteed to work smoothly through all environmentally realistic conditions, including passing over all kinds of surfaces; also, working through man-made anti-navigation attacks that include signal spoofing, signal jamming, or any kind of signal disturbing.
A multi-sensory navigation system can compensate for the deficiency of single sensor signals and can improve the reliance on the whole integrated navigation system [1]. Therefore, having a multi-sensor navigation system equipped on an aircraft is going to be a must in future air navigation; especially as much as anti-navigational systems are being developed and widely used. This kind of system can support navigation missions of civil or commercial UAVs, same as inside crowded cities or inside low GNSS coverage areas. A standalone inertial measurement unit (IMU) navigation system's performance is not efficient enough to track the exact position without having a big amount of error. This error is usually a location estimation error accumulated over time [2]. Visual aerial information can be obtained by using a low-budget light camera mounted on an aircraft, which can support the sensor fusion algorithm with regional data estimations. As visually similar objects may have varying colors and visual textures based on lighting conditions affected by weather and daytime, many challenges can present, including extreme variations in light intensity, object visibility, and complex textures. This kind of navigation system requires having a robust algorithm that can recognize the stored data despite the change of orientation and altitude, and work through different changeable atmospheric, environmental, and weather conditions. This includes variations in light intensity, visibility, and complex ground textures.

In some recent studies geometric features-based algorithms such as speeded-up robust features (SURF) [3] and scale-invariant feature transform (SIFT) [4] have been utilized in autonomous navigation for ground robotic vehicles [5], and Extended Kalman filter based implementation of vision-aided GPS-denied autonomous UAV Inertial Navigation System [6]. A vision-based approach for position and orientation estimation of UAV using EKF has been presented and improved later with particle filter as in [7, 8].In their approach, digital elevation maps (DEMs) and video recordings from a camera mounted on a UAV were used for stereo analysis, which showed better results than only INS navigation systems.

In some other studies, an approach of passive vision-based navigation has been successfully applied to micro-UAVs for indoor robotic applications as in [9, 10]. Whereby studying this kind of literature, we can notice that it represents some academic approaches to design algorithms suitable for long endurance, and low to medium-altitude outdoor environments. Some other related approaches had a navigation algorithm using limited predefined landmarks for specific locations (Small towns, rural buildings..etc) as waypoints. Whereas the dead-reckoning process was used to navigate the UAV between two waypoints, where the landmarks were detected using a scale, lighting, and rotation invariant image processing algorithm. Then the algorithm could estimate the current position, the orientation of the UAV, and the path to the next way-point approximately. The entire process was simulated using real and simulation aerial imagery [11]. The preliminary version of the proposed image processing-based navigation approach was presented at [12].

## 1.2 Objective of the Thesis

This study aims to develop an image processing-based adaptive multi-sensor fusion system to support accelerometer sensors and Global Navigation Satellite System (GNSS) based sensors for navigation in UAVs. The main purpose of having this combination of signals is to enhance the accuracy of the UAV navigation positioning estimation and the reliability of the navigation signals sensor fusion output, which will be robust enough to estimate precise positioning information despite the loss of one of the measurement signals or having instance bad measurements during the flight.

## 1.3 Hypothesis

Multiple methods can be utilized for aerial image processing and sensor fusion depending on the types of both measurement sensors and the acquired data. In this thesis, we propose a methodology to use previously scanned aerial map images' feature points as training data, that will assist the aircraft's live navigation system to operate by comparing live processed aerial images' feature points with the previously reserved feature points data, where the stored data is supposed to include the entire flight mission region, especially for flying over land missions. In the proposed method four combinations of scale and rotation invariant interest-point selection-based image processing approaches are utilized. The selected interest points are all together analyzed with the ground truth landmark locations. Then, the image processing position estimation is acquired. Image processing-based estimations are fused with the accelerometer sensor and the GNSS-based sensor values. Two adaptive Kalman Filter-based multi-sensor fusion methods are used in the fusion step of the proposed method. The proposed image processing and sensor fusion methods combinations results are all together compared to show the most accurate combination suitable for use in the autonomous navigation of UAVs.

## 1.4 Structure of the Thesis

The rest of this thesis is arranged as follows: In the next section, the essentials of the used image processing algorithm, the Kalman Filter algorithm, and its adaptive programming will be briefly explained. In Section 3, the proposed method is introduced. The simulation results of the proposed method are summarized in Section 4. Finally, the results and discussions are presented in the last Section.

# 2
# RELATED WORKS

## 2.1 Features Detection in Image Processing

When researching the topic of feature point detection in images, we can find SURF and SIFT algorithms as well-known reliable solutions compatible with a variety of image environments and conditions.

The Scale-Invariant Feature Transform (SIFT) was invented by David Lowe in the year of 1999 [4] and is specified to deal with affine transformations, image rotation, viewpoint, and intensity change in the image matching features. The SIFT algorithm has four main algorithm steps. In the first standard step, an approximation of Laplacian of Gaussian (LoG) which is called the Difference of Gaussian (DoG) operator is used to estimate the scale-space extrema, then it will be compared to find the difference of each image. Next, Using the DoG operator the keypoint candidates are filtered and enhanced by removing the low contrast points at various scales of the processed image, which will lead to what is known as keypoint localization. Then, using 16×16 neighborhood pixels located around each detected feature a keypoint orientation is calculated using the local image gradient. At last, to examine the local image descriptor for each detected keypoint a keypoint descriptor is created by using the image gradient magnitude and orientation and by using a 4×4 spatial grid including 8 gradient angle histograms [13]. Then, a feature vector of 128 values is extracted by focusing on the local image region around the keypoint. Where the following Equation represents the applied convolution of the difference of two Gaussian, computed at different scales, for an image $I(x, y)$.

$$D(x, y, \sigma) = (G(x, u, k\sigma) - G(x, u, \sigma)) * I(x, y) \tag{2.1}$$

Where G refers to the Gaussian function, k$\sigma$ and $\sigma$ are the difference of nearby scales and $I(x, y)$ is the image [14]. As SIFT descriptor is based on the Difference of the Gaussian detector it has the property of being invariant to image transformations [15]. The process of this algorithm has both positive and negative effects on real-time

applications as it is highly efficient in terms of quality, but it's relatively slow in detecting, processing, and matching images' features [4].
While for Speeded-Up Robust Features (SURF) algorithm it was first introduced in 2006 by Herbert Bay, Tinne Tuytelaars, and Luc Van Gool; where it is usually used as a feature detector for many objectives such as object recognition, classification, and image registration. Its algorithm uses a slightly different way of detecting features than SIFT algorithm. Where SIFT algorithm creates image pyramids of layers, then it takes the difference of filtered layers using Gaussian of increasing sigma value, while in the SURF algorithm, the created pyramid includes a stack of images of the same resolution without down-sampling for higher levels. Also, the SURF algorithm filters the stacks using a box filter of second-order Gaussian partial derivatives; as it utilizes the integral images which accelerate the processing of rectangular box filters [16]. SURF method uses the Hessian matrix-based blob detector to find key points; also it uses wavelet responses in both horizontal and vertical directions, which are supported with suitable weights to assign the orientation. This can be studied in detail in [3] and briefly in [14]. The following equation calculates the Hessian matrix in point $x$ with the coordinates of $(x, y)$ at scale of $\sigma$ :

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{2.2}$$

Where H represents the Hessian matrix,$L_{xx}(x,\sigma)$ convolution of Gaussian second-order derivative with the image $I$ in point $x$, the same applies for $L_{xy}(x,\sigma)$ and $L_{xy}(x,\sigma)$ [14].
For the SURF descriptor as it is based on the Fast-Hessian detector, it has the property of being resilient to image rotation and scale invariant. In addition, it has a big advantage in computing and matching features; where it depends on local gradient histograms and uses integral images to accelerate the processing speed [17]. Nevertheless, we can notice that there is a trade-off between processing speed and matching quality between algorithms; so, there is no best method for all deformations if we include illumination and blur deformation effects too. Since each of these algorithms may have the best results for specific cases while may have lower quality matching results for others, which can be seen in the comparison results of (Luo, Juan & Oubong, Gwun. (2009)) [16].

## 2.2 The Kalman Filter

Kalman filtering methodology briefly, is a product of using knowledge and familiarity with statistics and control theory to utilize measurement observations to estimate the desired unknown system state or variable. These measured observations mostly include statistical processing noise and inaccuracies over time. The filtering method was named after Rudolf E. Kalman, who had a major role in the development of the filter around the year 1960. The Kalman filter method has been vital in the applications of the navigation systems used in aircraft, UAVs, missiles, and spacecraft.

Mainly, the Kalman filter algorithm consists of two stages: prediction and update [18] (Kim & Bang, 2019). Where based on linear dynamical systems in state space format the filter predicts new system states using previous iteration system states while putting the current noise into consideration through the calculations. So, the evolution of the state from time $k-1$ to time $k$ can be presented as:

$$\hat{x}_k^- = F\hat{x}_{k-1}^+ - Bu_{k-1} + w_{k-1} \tag{2.3}$$

$$z_k = Hx_k + v_k \tag{2.4}$$

Where $F$ refers to the states transition matrix, the state vector is denoted by $x$, while the input u control matrix is denoted by B. while H is the measurement function that interrelates the current state $x_k$, to the current measurement $z_k$, while $w_(k-1)$ and $v_k$ are process model's Gaussian noises and the measurement model with covariance Q and R, respectively. (Kim & Bang, 2019) [18]

Once the prediction stage is done; both Kalman gain and updated state estimates are estimated in the update stage, where the internal navigator model state prediction and predicted error covariance can be represented as the following equations:

$$\hat{x}_k^- = F\hat{x}_{k-1}^+ - Bu_{k-1} \tag{2.5}$$

$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q \tag{2.6}$$

Where, $P_k^-$ is predicted error covariance, The hat operator $\wedge$, is an estimate of a variable, where $\hat{x}$ is an estimate of $x$. The superscripts $-$ and $+$ denote predicted (prior) and updated (posterior) estimates (Kim & Bang, 2019) [18]. While update stage can be modeled as:

$$\tilde{y} = z_k - H\hat{x}_k^- \tag{2.7}$$

Where $\tilde{y}$ is the Measurement residual which will add up on each time step to the previous estimation after being multiplied to Kalman gain as will be shown next, where $H$ Matrix is set to be (6 by 6) matrix to convert predicted states matrix dimensions to the desired (6 by 6) states. Kalman Gain's value will be updated as the following equation:

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

Then the State Estimate value will be updated based on Kalman gain as the following equation:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k \tilde{y} \tag{2.9}$$

Then the Error Covariance will be updated as well as the following:

$$P_k^+ = (I - K_k H_k) P_k^- \tag{2.10}$$

Whereas we are working on a simulation model in this algorithm $M = 100$ Monte-Carlo runs were conducted; as a single run is not sufficient for verifying the statistical characteristic of the filtering result, because each sample of noise differs whenever the noise is sampled from a given distribution, and therefore, every simulation run results in different state estimate. The repetitive Monte-Carlo runs enable us to test several different noise samples for each time step (Kim & Bang, 2019) [18].

## 2.3 Adaptive Kalman Measurement Programming

In this thesis, two adaptive Sensor Fusion methods will be represented, as we are going to have inputs signal from different sensors for the same states. In reality, two sensors measurements will not occur instantaneously at the same instant due to distinct sensors' output frequencies differences; So, Usually, they are dealt with as separated measurements or in other cases, one of them will be considered as a delayed measurement, by putting the delay time difference in consideration during the data

fusion.
As we are dealing with multiple sensors providing the same states at the same time "theoretically" we have to deal with the issue of identifying the one accurate and the one inaccurate sensor measurement without completely discarding the information coming from any of these sensors measuring the system, including the imprecise ones. In some cases, a complementary filter is used for correction [19], Where it applies a low pass filter on one of the measurements and a high pass filter on the other measurement; depending on the value of trust which is allocated for each of the sensors' measurements, where this trust value is based on sensors' previously known error occurrence probabilities. This method may not be so efficient in case of dealing with image processing-based navigation as bad measurements may occur unpredictably. Obtaining an adaptive measurement can be possible by calculating the Euclidean distance between each of the sensor's measurement coordinates and the current inertial navigator's estimation coordinates, the result which has less distance value is considered more accurate and the farther is considered less accurate. In addition, a check condition can be added that can take the aircraft's performance limits into account (for example possible speed limits and acceleration), which can be used to make sure which measurement is more reasonable. Another approach solves this by using the "Gating" technique, which aims to avoid the detected corrupted measurements, and rather, it depends on the last navigator estimation with regard to the gate limit value, which will decide about the current Kalman filter estimation update, which can be studied in (Roger R Labbe Jr 2020) [20].

# 3
# THE PROPOSED METHOD

## 3.1 Data Acquisition and Map Stitching

To conduct the proposed method, it was required to utilize the same camera setup to record two different training data flights, each flight was recorded from a different height. This recorded data will be processed and stored to be used in the live algorithm as a location reference, the training data processing part includes stitching each of the training data sets to create two aerial mosaic maps. Where MATLAB's [21] SURF features and descriptors extraction and matching functions, also MSAC's **estimateGeometricTransform** functions have been used to stitch images and create the mosaic's big map for each of the training data sets. Having low-quality stadium background required using very low thresholds in feature detection, as at 10m height images looked blurry. (Where M-estimator SAmple Consensus MSAC is a variant of the Random Sample Consensus (RANSAC) algorithm) [22]. Also in this thesis, the AGISOFT program [23] was used to generate Orthomosaic maps from stitched image tiles, which gave better results as will be shown in the Experimental Results section. These mentioned tiles have been previously processed by the Matlab image processing toolbox. To further study the used tool functionality Orthomosaic & DEM generation (without GCPs) topic can be researched in this documentation page [24].

Next, The Stitched maps Image feature points and their descriptors have been extracted and stored, which will be used in test flight simulations in the features matching part. In this thesis,**4** types of feature points and descriptors extraction algorithm has been represented to show the variation of different combinations' results. In the first method, **only SURF** feature points and descriptors have been extracted; while in the second **SURF feature** points have been detected and **SIFT descriptors** has been extracted, in the third **SIFT feature** points has been detected and **SURF descriptors** has been extracted, which has shown better results in matching features as will be shown in the experimental results section. In the fourth method **only SIFT** feature points and descriptors have been extracted. Where the used parameters for each feature extraction method are shown in the Experimental results

part.
At the end of this part before applying the Test flight algorithm, all Ground Control Markers pixel coordinates are been extracted from both Mosaic images manually as pixel geometric coordinates; which will be helpful as a reference in converting between pixel geometric coordinates to real coordinates through the image processing algorithm, So, all land Markers real coordinates locations are known in both Mosaic Pixel geometric coordinates and also Real metric (x-y) coordinates before the implementation of the test flights. Knowing this combination of coordinates can reduce the error in image distance estimation between two different points in case there is any bad stitching in the mosaics.
Some manual calculations are required to be performed after the creation of the mosaic maps, which aim to calculate the ratio change between pixels per meter after applying the stitching algorithm in training data. Due to the nature of MSAC algorithm, the ratio of pixel-meter has been changed by a small ratio and the size of the created mosaics has shown less size than what it should be if photos were stitched perfectly without any scale reduction, which has been added to consideration in live image processing "altitude" estimations, as can be seen in the next section.

## 3.2 Live Image Processing Algorithm

One of the main purposes of our proposed method is to ensure flight safety by supporting the aircraft flight with previously recorded aerial data, which will help the aircraft's flight navigation system with locating and detecting its true position as accurately as possible. So, live image processing is the process done during the real flight of the drone where image processing is done for the instantly taken aerial images. Then comparing these live extracted images' features with previously stored aerial data. In this thesis, two test flights have been conducted, and the created simulation test flight scenarios are explained in detail in the Experimental results section.
The following diagram in Figure 3.1 summarizes this processing algorithm explained in the next subsections:

### 3.2.1 Live Images Features Detection and Matching

This part of the algorithm is responsible for linking the test flight's live visual input with the stored data, which have already been recorded and processed previously in training flights. Through the test flight, aerial live images are going to be captured at a specific rate. For each recorded aerial image valid feature points and their descriptors are extracted. Then, using **matchFeatures** MATLAB function Strongest features matching is applied between the stored feature points pool and the live

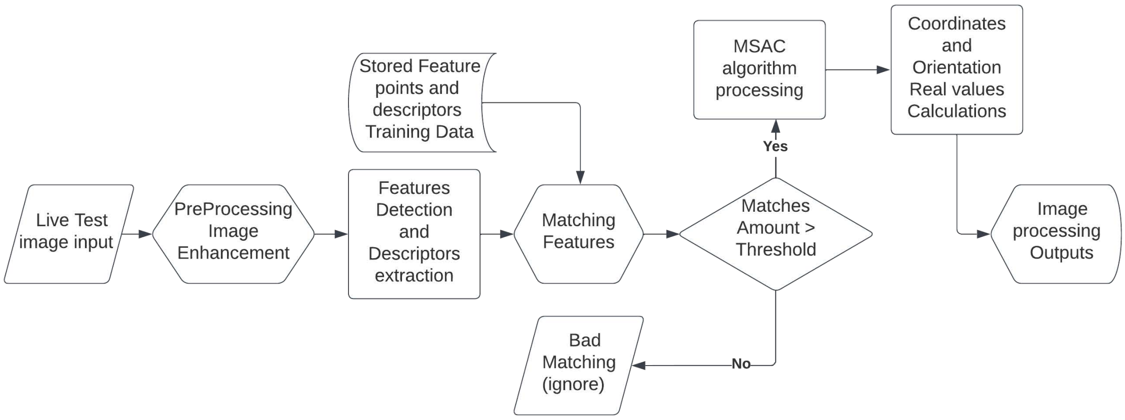


**Figure 3.1** An Overview of Image Processing Part of The Proposed Method

image features. Where, MATLAB default parameters have been used for this function, which can be studied in this MATLAB documentation [25]. Then, **"estimateGeometricTransform"** MATLAB function has been used after finding the best mosaic match that fits with the live image. The used function returns a 2-D geometric transform object; where this object maps the inliers between two different matched points. As result, this function excludes outliers using the M-estimator SAmple Consensus (MSAC) algorithm, which is a variant of the RANSAC algorithm. The used function parameters are **'similarity'** parameter which is used to set the Minimum Number of Matched Pairs of Points. Also, **'Confidence'** parameter is used as it represents the Confidence of finding the maximum number of inliers, where increasing this value improves the robustness of the results at the expense of additional computations. In addition to that, **'MaxNumTrials'** parameter was used, which stands for the maximum number of random trials for finding the inliers, where increasing this value improves the robustness of the results at the expense of additional computations as well.
As the RANSAC algorithm function is an iterative function that filters out the less possible matches (outliers) from the more possible matching pairs (inliers) of feature points; Results may not be the same between runs because of the randomized nature of the MSAC algorithm [22].

### 3.2.2 Live Images Pixel Coordinates Recovering

This part is responsible for calculating the aircraft's location in the mosaic map's pixel coordinates. Where the aircraft's location is assumed to be in the center of the mounted camera's view. As we put into consideration flight tests camera orientation is vertical.
To recover the live image's scale, orientation, and location pixel coordinates, we used the image geometric transformation matrix **"tform"**, Which is the output of the MSAC

algorithm that was used in the previous part. This information has been extracted from the inverse matrix of **"tform"** which we called ($Tinv$) which looks as shown in 3.1. As we are comparing the live captured image's feature points with two mosaic maps, we will get two coordinate estimations as output, Where each of these mosaic maps is recorded from different altitudes.

$$Tinv = \begin{bmatrix} S.C_\theta & -S.S_\theta & 0 \\ S.S_\theta & S.C_\theta & 0 \\ tx & ty & 1 \end{bmatrix} \tag{3.1}$$

Where $S.S_\theta$ and $S.C_\theta$ expressions can be explained as:

$$S.C_\theta = S.cos(\theta) \tag{3.2}$$

$$S.S_\theta = S.sin(\theta) \tag{3.3}$$

Where $tx$ and $ty$ are translations over x and y coordinates, respectively. while $S$ in equations 3.2 and 3.3 is the geometric scale transformation value that will be recovered, and $theta$ is the geometric rotation angle transformation value that will be recovered as well. The recovered scale and rotation values are can be estimated in the equations 3.4 and 3.5:

$$RecoveredScale(S) = \sqrt{(S.S_\theta).(S.S_\theta) + (S.C_\theta).(S.C_\theta)} \tag{3.4}$$

$$RecoveredRotation(\theta) = atan2(S.S_\theta, S.C_\theta).180/\pi \tag{3.5}$$

The recovered (theta) value found in 3.5 is useful to know the orientation and movement direction of the UAV in that instance of time. In addition, we can calculate the altitude approximately using the scaling value found in 3.4. These steps can be studied with simple visual examples in [26]. As we already know the height which mosaic map images are taken from. So, by doing the proportional multiplication we get:

$$Altitude_{30M} = 30/(RecoveredScale_{30mMosaic}) \tag{3.6}$$

$$Altitude_{10M} = 12.9/(RecoveredScale_{10mMosaic}) \tag{3.7}$$

And by taking the average of the estimated altitudes we get the final altitude estimation.
Finally, to find the live image's pixel coordinates the MATLAB function **"transformPointsForward"** has been utilized using the parameters of the 2-D geometric transformation matrix "**tform**" and using the center view coordinates of the mounted camera. As result, we get the geometric center pixel coordinates of the live test image location in each of the mosaic map images, for better understanding, we name that point (**C**).

### 3.2.3 Converting from Mosaic Pixel to Real Meter Coordinates

This part of the algorithm has the purpose of converting the digital information which we received before into real-world coordinates, or technically into simulation-world (X-Y) coordinates. All of the fixed markers' real-world and geometric pixel coordinates also the (pixel to meter) conversion ratio values are calculated and stored during the processing of the training data for each mosaic map, which will have a major effect in this section.
Firstly, using the geometric pixel coordinates of points (C) we find the closest 3 ground markers. Let's call these Markers (A, B, and D). Then we calculate the geometric pixel distance between point (C) and each of (A, B, and D). Secondly, the calculated geometric pixel distances are converted into meters using the pixel-to-meter conversion ratio values. Then, a MATLAB **Symbolic** system is created to find the actual coordinate values of (C) point $(x_{real(C)}, y_{real(C)})$.By solving the Euclidean distance equations of 3.8 and 3.9 in this Symbolic system we will get two solutions as we have two unknowns. In other words, we are looking for the possible two intersection points' coordinates far from Marker A and Marker B's real meter coordinates by the previously calculated meter distances. As shown in Figure 3.2.

$$AC_{MeterDistance} = \sqrt{(x_{real(C)} - x_{real(A)})^2 + (y_{real(C)} - y_{real(A)})^2} \tag{3.8}$$

$$BC_{MeterDistance} = \sqrt{(x_{real(C)} - x_{real(B)})^2 + (y_{real(C)} - y_{real(B)})^2} \tag{3.9}$$

Equation 3.8 represents the meter distance between Marker (A) meter (x-y) coordinates and the unknown center point (C) meter coordinates which we want to calculate. While equation 3.9 represents the meter distance between the Marker B meter (x-y) coordinates to the unknown center point (C) meter coordinates which we want to calculate.

To know which solution is correct we use the calculated two possible solutions of **(C)** point; we use the metric distance between the point (C) and Marker (D) which we converted from geometric pixels meters before (we call this distance as $D_x$), Then we calculate the distance between each of the two possible solutions points' to marker (D) (let's name these distances as $D_1$ and $D_2$) which are calculated in the following equations 3.10 and 3.11.

$$D_1 = \sqrt{(x_{solution1} - x_{real(D)})^2 + (y_{solution1} - y_{real(D)})^2} \tag{3.10}$$

$$D_2 = \sqrt{(x_{solution2} - x_{real(D)})^2 + (y_{solution2} - y_{real(D)})^2} \tag{3.11}$$

Then, to find the final accepted solution we do a distance comparison, the solution point **($D_1$ or $D_2$)** which is far from Marker (D) coordinates by a metric distance value **"closer"** to ($D_x$) will be considered as a true solution, while the other solution will be ignored. In mathematical expression:

**if $|D_1 - D_x| < |D_2 - D_x|$ is true Solution1 is accepted**
**else if $|D_2 - D_x| < |D_1 - D_x|$ is true Solution2 is accepted**

Where the same algorithm is applied for each of the mosaics markers' locations and the average of the two results is considered as the last (x-y) meter estimation for the drone location in real simulation coordinates.

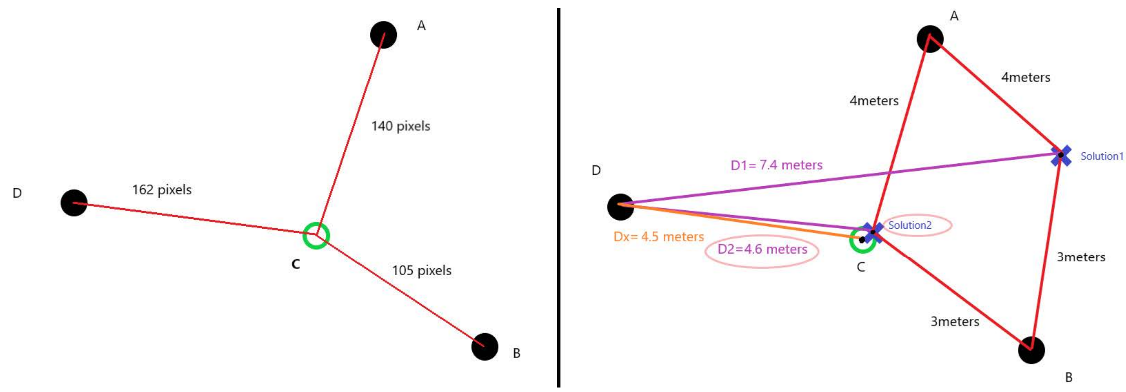


**Figure 3.2** An Approximate Example of Converting Pixels Coordinates to Metric Coordinates(Each 35 Pixels Distance is Converted Into Approximately 1 Meter). Green Spot is The Geometric Pixel Estimation, Black Spots are Ground Markers, and Blue (X)s are The Possible Solutions

## 3.3 Kalman Filtering Algorithm

A multiple-sensor navigation system configuration can rectify every single sensor's deficiency and limitations. In this thesis, we are dealing with multiple sensor input measurements, including IMU measurements, noisy GNSS signal, and image processing coordinates estimation signal; A balancing measurement input weight value (Gain) must be calculated to balance how the algorithm selects the more accurate input measurement. GNSS noisy signal was acquired by using the real drone flight coordinate with the addition of 10 meters zero mean random Gaussian noise. Two measurement signals fusion methods are introduced in this thesis as will be shown in Sections (3.3.1) and (3.3.2).

### 3.3.1 ONE KMEAS GAIN Measurement Method

In this proposed method **Kmeas** gain has been calculated, which stands for the Measurement gain. We used this updating gain value to balance the trust between both input signals (GNSS and Image processing coordinates measurements) for every estimation cycle. **Kmeas** gain will be updated in each cycle based on the calculated measurements error proportion as shown in 3.12. The error values can be found by calculating the Euclidean Norm (length) value of the difference between each of the measurements and the Kalman inertial navigator's current estimated value for each estimation, as shown in equations 3.13 and 3.14, The entire proposed method is illustrated in Figure 3.3.

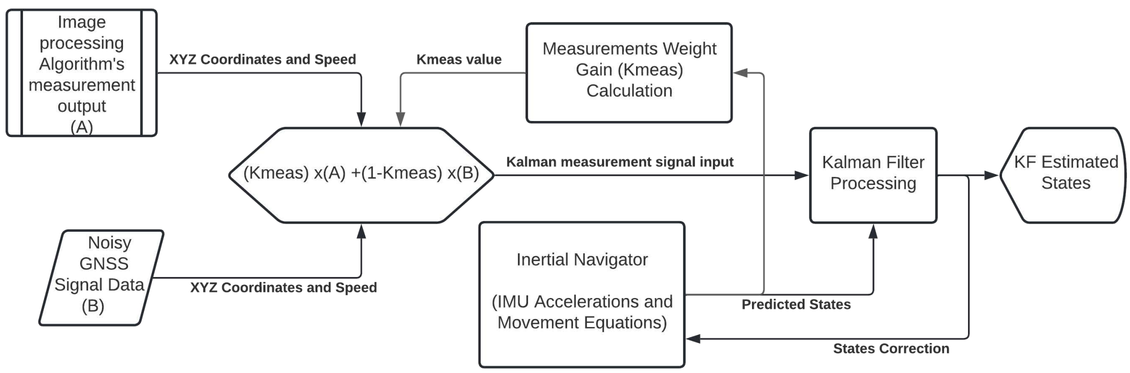


**Figure 3.3** (1Kmeas) Gain Measurement Proposed KF Method

$$Kmeas_k = \frac{(ErrGNSS_k)}{(ErrGNSS_k + ErrImgProcc_k)} \tag{3.12}$$

Where $ErrGNSS_k$ and $ErrImgProcc_k$ are GNSS and Image processing measurement signals error (its difference from the inertial navigator prediction). Which can be calculated using the equations 3.13 and 3.14.

$$ErrGNSS_k = norm(GNSSNoisy_k - \hat{x}_k) \tag{3.13}$$

$$ErrImgProcc_k = norm(ImgProccMeas_k - \hat{x}_k) \tag{3.14}$$

Where $\hat{x}$ is the current Kalman coordinates prediction (current navigator states prediction), and "GNSSNoisy" stands for the current GNSS noisy measurement input, which has a maximum of 10 meters of noise. While "ImgProccMeas" is the current image processing coordinates measurement input. As, by finding the Euclidean Norm (length) of the difference between the Kalman filter's predicted states and the measurement signals we can find an approximate error value for each of the measurement inputs, such as "ErrGNSS" and "ErrImgProcc". Then, using $Kmeas$ value we found in 3.12 we calculate the total input measurement as shown in Equation 3.15:

$$z_k = (Kmeas * ImgProccMeas_k) + ((1 - Kmeas) * GNSSNoisy_k) \tag{3.15}$$

Where $z_k$ is the Kalman filter's total measurement input signal, or we can call it the fused measurement signal. In this method, we update the fused measurement error covariance matrix "**R**" based on the amount of possible error distribution added from the combination of different measurements. This updating will be related directly to each measurement's percentage effect in the fused measurement, which is expressed by the "$Kmeas$" value.

In this thesis, we will have 3 different values of the standard deviation of **measurement noise**. These values are related to which measurement is used, they are symbolized as $sig_{mea}$ , $sig_{meaG}$ and $sig_{meaI}$ . Where $sig_{meaG}$ is the standard deviation of GNSS **measurement** noise used in $RG_k$, where $RG_k$ is the GNSS measurement error covariance matrix. And $sig_{meaI}$ is the standard deviation of Image Processing **measurement** noise used in $RI_k$, where $RI_k$ is the Image processing measurement error covariance matrix. $sig_{mea}$ is standard deviation of the **fused measurement** noise which will be used as initial value for "**R**" matrix.Then the "**R**" matrix will be updated for each measurement As shown in equation 3.16.

$$R_k = (Kmeas_k * RI_k) + ((1 - Kmeas_k) * RG_k) \tag{3.16}$$

### 3.3.2 THREE GAINS Measurement Method

In this proposed method we aim to balance the effect of the multi-sensors signal on the fused measurement signal by calculating a separated Kalman gain for each measurement **(K1)** and (**K2**), then we calculate a "**Kmeas**" measurement fusion gain using the values of the calculated gains **(K1)** and (**K2**) as shown in equation 3.19.

Where the calculated "Kmeas" gain's work is similar to **One Kmeas** gain method, which is explained in the previous subsection. The same Error covariance matrices values have been used for (**RI**) and (**RG**) matrices, but the way of choosing the measurement is going to be based on the updated Kalman gains **(K1)** and (**K2**) for each measurement.

By using the same shared (**P**) predicted error covariance matrix, each of the measurement's Kalman gain is updated based on its (**R**) covariance matrix separately. Then, the measurement fusion gain is calculated based on the amount of trust that Kalman gains (**K1**) and (**K2**) have for each measurement, as shown below:

$$K1_k = P_k^- H_k^T (RG + H_k P_k^- H_k^T)^{-1} \tag{3.17}$$

$$K2_k = P_k^- H_k^T (RI + H_k P_k^- H_k^T)^{-1} \tag{3.18}$$

$$Kmeas_k = \frac{K2_k}{(K1_k + K2_k)} \tag{3.19}$$

So, the measurement input value and the new measurement's Error covariance matrix will be estimated finally as the following equations 3.20 and 3.21:

$$z_k = (Kmeas_k * ImgProccMeas_k) + ((I - Kmeas_k) * GNSSNoisy_k) \tag{3.20}$$

$$R_k = (Kmeas_k * RI_k) + ((I - Kmeas_k) * RG_k) \tag{3.21}$$

# 4
# EXPERIMENTAL RESULTS

In this thesis, MATLAB application (Version R2022a)[21], AGISOFT (trial version)[23], and Unity game engine's (LTS 2019.4.32f1) [27] free version's based flight simulator programs have been used to perform and implement all of the data acquisition and to design and implement Kalman filter and image processing algorithms. In the simulation part, the Google Earth program's aerial image [28] was scientifically utilized as an aerial image source for the specified simulation flight region as shown in figure 4.1 and 4.2.

## 4.1 Data Acquisition and Map Stitching Results

In real-life conditions, aerial data acquisition can be accomplished using high-quality satellite images, as well as using scout drone aerial scans. Whereas in this approach, training and test data were recorded in a simulated 3D environment. Two training data sets and two test data set has been recorded to obtain the required data, that will help the aircraft identify its location through the live image processing algorithm. By using the UNITY game engine-based flight simulator; the simulation flights have been recorded over a 3D environment (including aerial images and their coordinates). Where the stadium surface included colored markers with specific lengths and shapes, also including changeable locations to increase the image processing challenge for different flight scenarios, which will be explained in the Experimental results part. All these markers' locations' "real" coordinates are known, which will be considered as a reference to determine the real location of the drone (the simulation view camera) through the algorithm. The designed simulation application allows maneuvers and changes in height and speed of the aircraft, also it allows recording the flight path and direct view images, which have been used in both training data and test data acquisitions. In the simulation environment, each of the boxes has a unique color (red, green, or blue).

Two training data flights were recorded over a 3D environment, where the recording

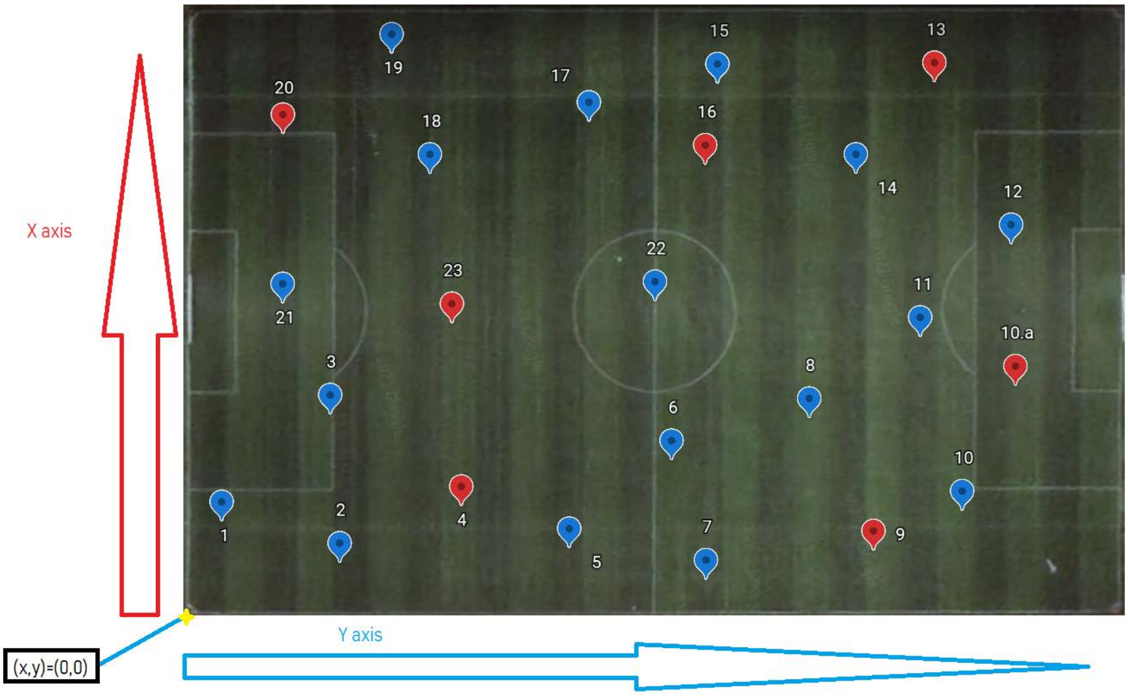


**Figure 4.1** Aerial Image of Stadium-Markers Locations, raster image taken From Google Earth Program [28]

3d environment's approximate total lengths are (104.83m×67.5m) and have 24 colored markers (1-meter squared size box each). Markers included 7 markers whose locations are changeable, to have it changed in test flights to increase the image processing challenge.

The used simulation Drone Camera's properties include having a diagonal field of view (DFOV) equal to 80 and (1280×720) pixels Camera Resolution.
The first training data flight simulation was at **30m** height, about 100 photos were recorded, and the camera movement speed was 10 m/s. while the image Recording frequency was fixed at 1Hz. MATLAB Image processing toolbox was used to stitch each horizontal scan separately to create 4 mosaic images, then the AGISOFT program was used to stitch the recorded data without ground control points; which showed better Mosaic results than MATLAB for this data set. The used AGISOFT stitching and mosaic-creating functions and options can be seen on this documentation page [24]. The resulting mosaic's raw pixel dimensions were about 2450×6564 pixels, where black unused sides were cropped to prevent image processing confusion.
The second training data flight simulation was at **10m** height, about 2000 photos were recorded, camera movement speed was 6 m/s. While the image recording frequency was fixed at 3 Hz. MATLAB Image processing toolbox was used to stitch each horizontal scan separately to create 14 mosaics, then used MATLAB again to stitch them all to create a full-scale mosaic for the stadium.

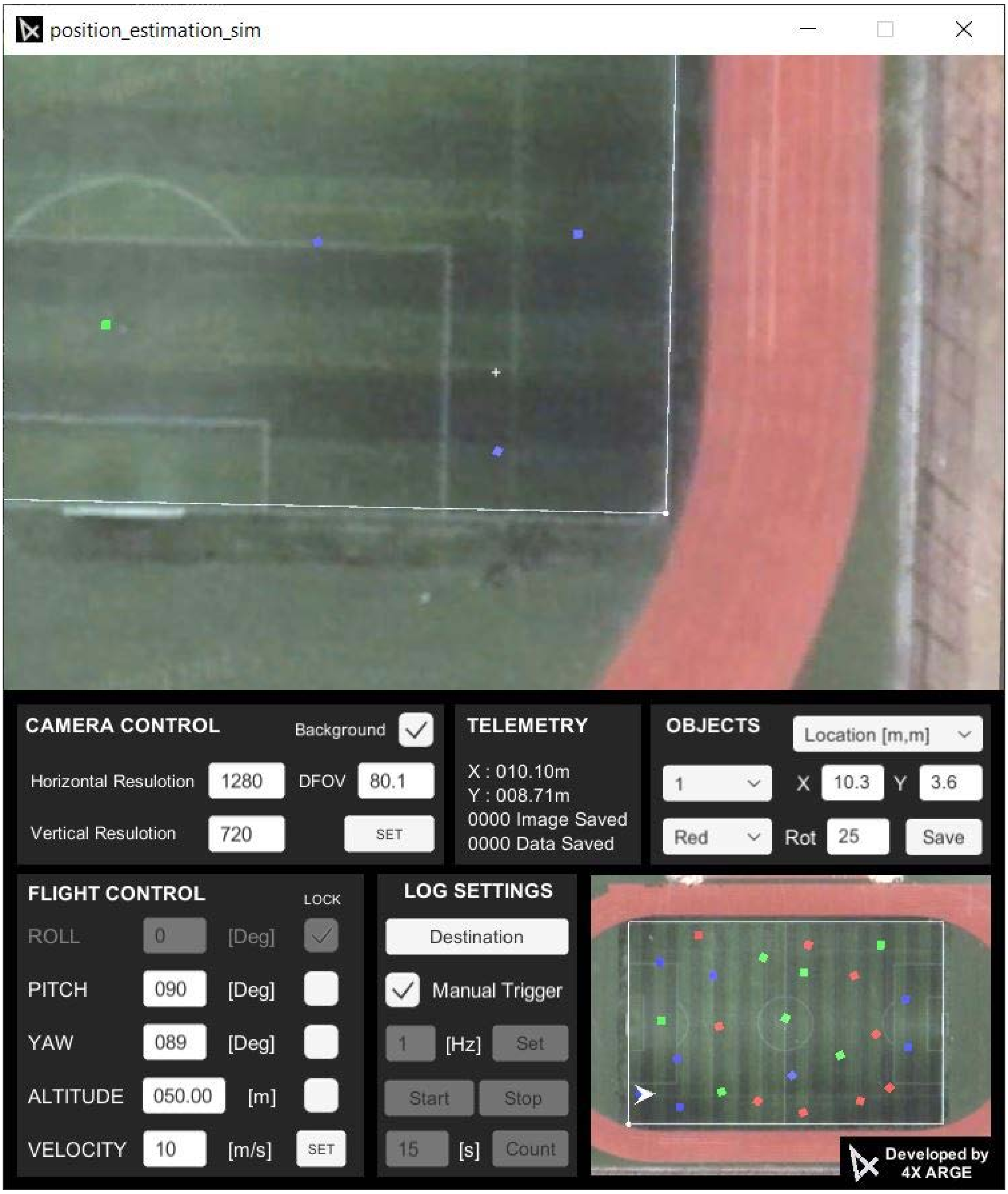


**Figure 4.2** The Interface of The Designed Unity Engine-Based 3d Environment Flight Simulator, The Background Image is Taken From Google Earth Program [28]

In this thesis, we have implemented 4 types of feature points and descriptors extraction algorithms on the same data. The number of features extracted using each of these 4 algorithm types is shown in Table 4.1. where **SIFT** feature points detection and **SURF** descriptors extraction have shown the best results in matching features as presented :

**Table 4.1** Stitched Mosaic Maps' Extracted Feature Points and Descriptors Numbers.

| Stitched Mosaic type | Mosaic (30m altitude) | | | | Mosaic (10m altitude) | | | |
|---|---|---|---|---|---|---|---|---|
| Detection and Extraction Algorithm type | SURF only Features and Descriptors | SURF Features and SIFT Descriptors | SIFT Features and SURF Descriptors | SIFT only Features and Descriptors | SURF only Features and Descriptors | SURF Features and SIFT Descriptors | SIFT Features and SURF Descriptors | SIFT only Features and Descriptors |
| Detected and Extracted Valid Feature Points and Descriptors number | 7860 | 8844 | 11410 | 5376 | 7850 | 8858 | 11357 | 5169 |

The used MATLAB functions for SURF-only feature detection and descriptor extraction is **detectSURFFeatures** where the parameter of **"MetricThreshold"** have been used, and its threshold value was specified to be 20, we had to lower its value as the stadium background image in the simulation was not high quality enough to detect features, the default threshold value was around 1000. For descriptors extraction part **extractFeatures** function have been used; which was set to extract SURF features in this method. While in the second algorithm, the used MATLAB functions for SIFT feature detection and SURF descriptor extraction is **detectSIFTFeatures** where the parameter of "**Sigma**" has been used and its value was set to be 1 in the 3rd method and 2 in the 4th method. "**Sigma**" stands for the sigma of the Gaussian, which is applied to the input image at the initial octave. Its values are usually in the range of (1 to 2). Lowering the sigma value helps in cases of blurry images based on MATLAB documentation. "**EdgeThreshold**" parameter was set to the value of 5, which is useful to filter out unstable edge-similar features in the image that may be sensitive to noise; where its value was reduced to gain more detected feature points from the blurry stadium background. Same as for "**ContrastThreshold**" which was set to 0.00333 value to decrease the filtered-out weak features in low-contrast regions of the images. Also "**NumLayersInOctave**" parameter value has been set to 1, which stands for the number of layers in each image octave, as the octaves' number is automatically computed based on the size of the image. This parameter allows us to detect bigger or smaller features in the image. For the descriptors extraction part, **extractFeatures** function was used which was set to extract **SURF** features in this method too.

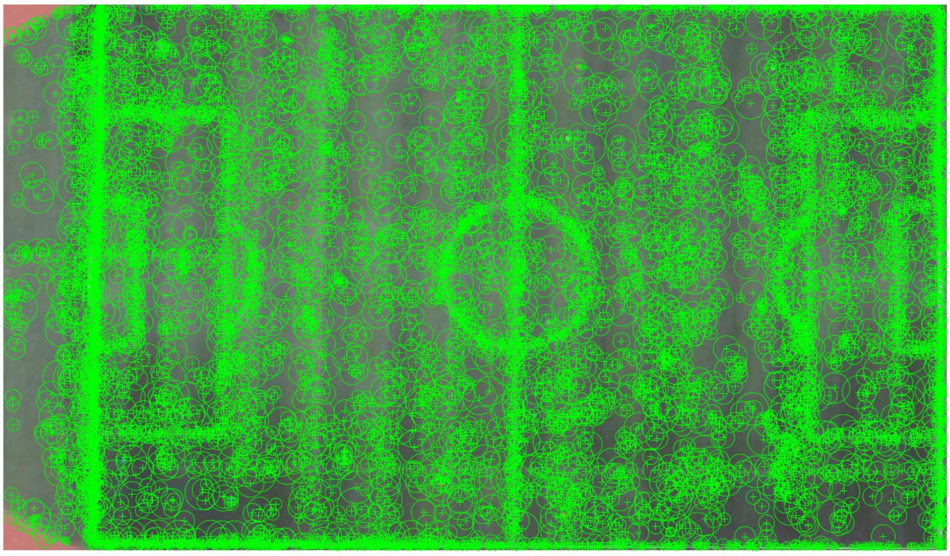

**Figure 4.3** 30 Meters Altitude Mosaic Extracted Feature Points and Descriptors

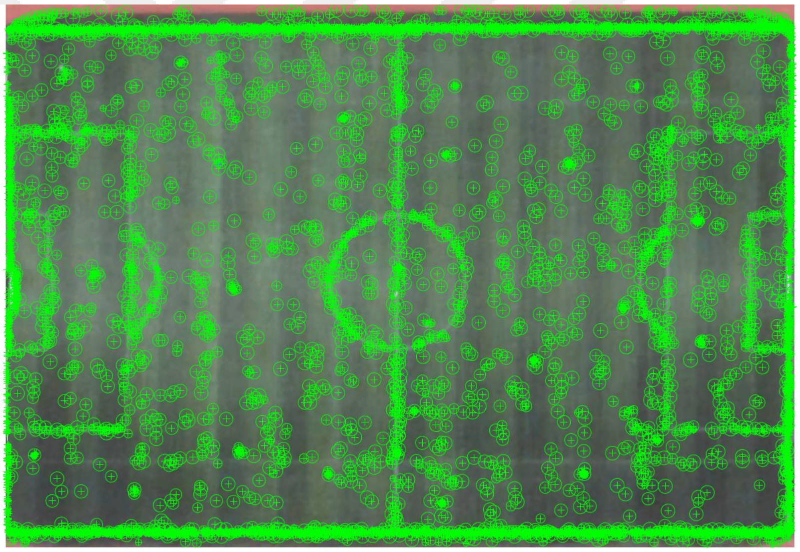

**Figure 4.4** 10 Meters Altitude Mosaic Extracted Feature Points and Descriptors

At the end of the data acquisition part, the manual meter-pixel calculations were done. it has been added to consideration in live image processing "altitude" estimations, as shown in the live image processing algorithm section. As result in **30m** height Mosaic, every 34.89 pixels stand for 1 meter in x-y simulation coordinates, and in **10m** height mosaic every 80.41 pixels stand for 1 meter in x-y simulation coordinates.

## 4.2 Live Image Processing Parameters and Results

For test simulation flights both live images and their real coordinates have been recorded. Two types of simulation test flight scenarios have been simulated in a 3D environment, and the recorded drone vertical images and coordinates have been stored to be used in MATLAB as test data sets. In the 1st scenario, the drone flies over the stadium area, which was fully included and processed in the training data, The stadium has some ground markers being moved to different locations to increase the challenge, while in the 2nd scenario, the drone flight included flying over some parts of the stadium which are not contained in training data, also the 2nd scenario included passing over some covered areas through its flights; in addition to doing a lot of direction changes. These challenges are added to show the effects of the fused image processing algorithm coordinates estimations in comparison to navigating using only GNSS signal and IMU, which will be explained in the next section. All the test flights were done at 20 meters altitude and the drone speed was set to 10m/s, recording frequency is 4Hz for the 1st scenario and 2Hz for the 2nd scenario.

In each test flight scenario, the live image processing algorithm included the detection of feature points and extraction of descriptors for each (image processing) method separately. The feature detection and extraction settings applied in training data are identical to the ones applied to the live flight algorithm.

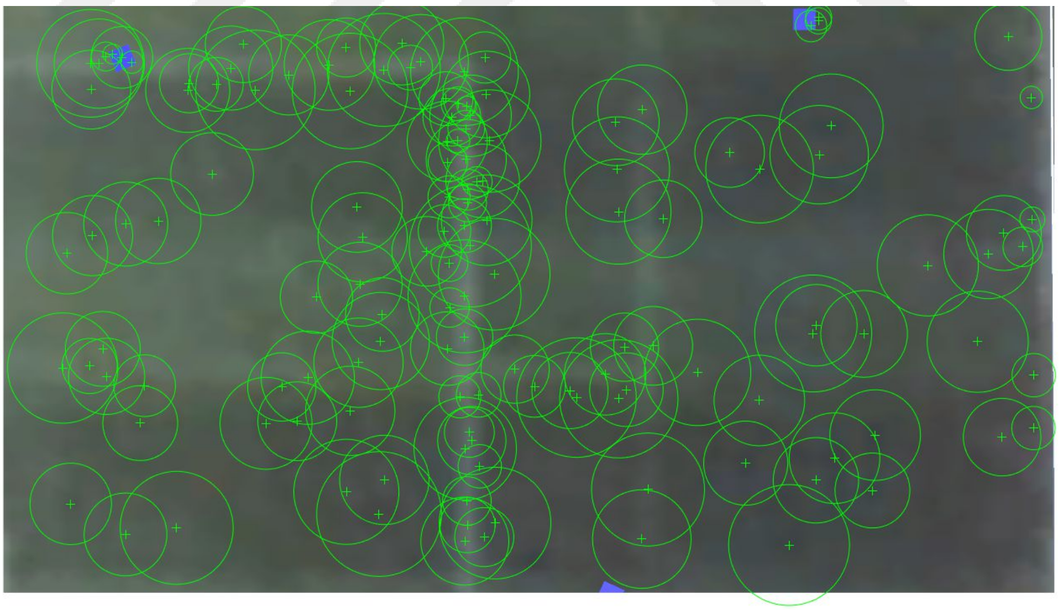

**Figure 4.5** Detected Feature Points and Descriptors of Live Test Image from Scenario1

The used parameters values in (MSAC) function are for **'similarity'** the value of two has been used, also **'Confidence'** was set to 99.9 from a maximum of 100. In addition to that, **'MaxNumTrials'** value was set to 2000 trials as well. The following photos in Figure 4.6 and Figure 4.7 show an example of matched SURF points for an instance of time with each of the mosaic stadium maps' feature points.

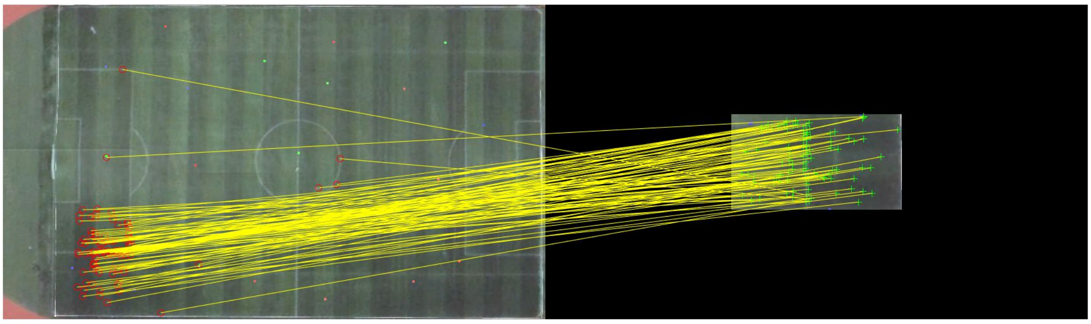

**Figure 4.6** Matched Feature Points with (Mosaic 30m)on The Left With a Test Data Image on The Right

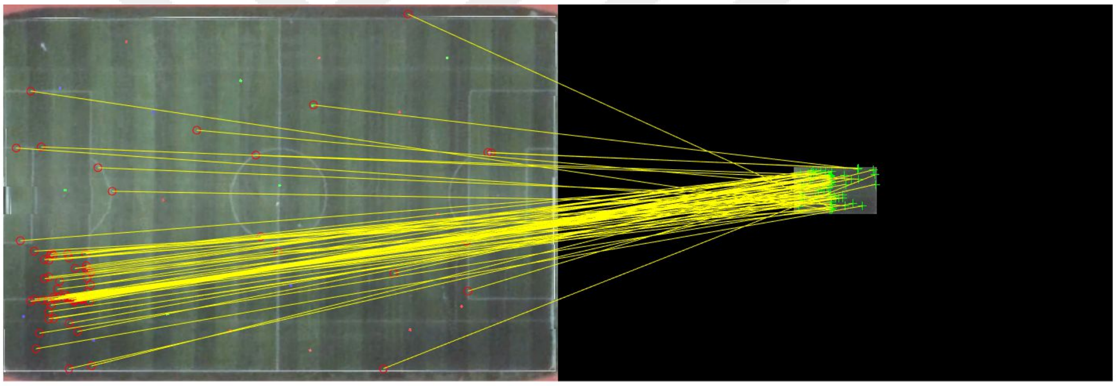

**Figure 4.7** Matched Feature Points with (Mosaic 10m) on the Left With a Test Data Image on The Right

## 4.3 Kalman Filter Algorithm Parameters

In this thesis, The state vector contains the three-dimensional position and velocity in the form shown below:

$$x_k = [p^T, v^T]^T$$

where $p = [p_x, p_y, p_z]^T$ is the position vector and $v = [v_x, v_y, v_z]^T$ is the velocity vector ,its elements are defined in x, y, z axes. For the One Kmeas method (shown in section 3.3.1) the used values for the standard deviations of measurements noise of $sig_{mea}$, $sig_{meaG}$ and $sig_{meaI}$ which are used in calculating the measurement noise covariance matrices used in the equations 3.16,3.17,3.18,3.20 and 3.21 are as the following:

$$sig_{mea} = [0.836, 0.836, 0.44, 1, 1, 0.2]^T$$
$$sig_{meaG} = [3, 3, 3, 4, 4, 4]^T$$
$$sig_{meaI} = [0.836, 0.836, 0.44, 1, 1, 0.2]^T$$

where $sig_{mea}$, $sig_{meaG}$ and $sig_{meaI}$ are explained at the end of (section 3.3.1), the values of $sig_{mea}$ are chosen based on the average values of 100 times test done on the measurement algorithm, where these values are related to the statistical probability of error associated with each of the measurement state estimates. The same idea applies to $sig_{meaG}$ which is related to the generated GNSS measurement positioning signal, whose measurement error may reach a maximum of 10 meters. For both (One Kmeas gain) and (3k gains) methods for each of the conducted Monte-Carlo runs the initial guess for the drone's initial state is set to be randomly sampled from a Gaussian distribution with a standard deviation of $[1, 1, 1, 0.3, 0.3, 0.3]$ states values that are in the same form of the state vector $\boldsymbol{x_k = [p^T, v^T]^T}$. While the (IMU acceleration) values are generated from the Gaussian distribution with a standard deviation of $[0.3, 0.3, 0.3]$ values applied on real data, which adds a random error value to the IMU acceleration measurement signal randomly sampled from a Gaussian distribution noise with the specified standard deviation. Similarly, For the Process Noise Covariance Matrix of the accelerometer sensor, standard deviation values of $[0.3, 0.3, 0.3]$ have been used.

## 4.4 Methods Results Comparison

To illustrate the test results the statistical results for each method's output will be shown in this part of this thesis. As mentioned in previous sections,100 Monte-Carlo runs have been conducted for each method and each test data set via the Matlab simulation. Then, the root means square error **(RMSE)**has been calculated for each of the estimated coordinate signals (X, Y, Z) of all runs, as it will be shown in the next figures for each method and test scenario separately. In addition to that, RMSE signals' means and variances have been calculated to show the general dispersion of the RMSE signal of the estimations for each method and scenario separately. as shown in the next tables 4.2 and 4.3.

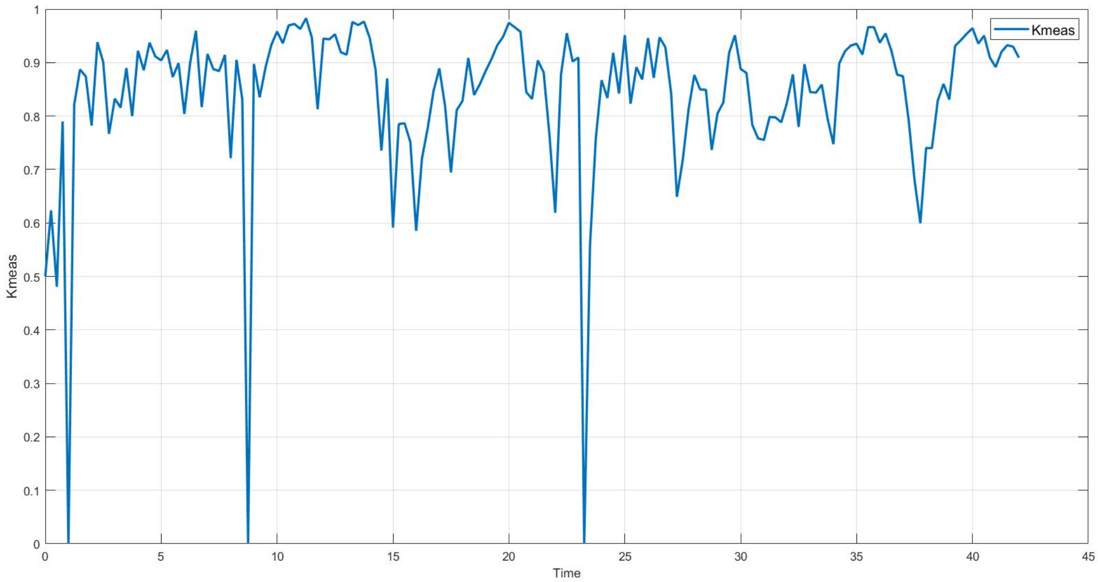


**Figure 4.8** (Kmeas) Measurement Gain values Over Time Used in 1kmeas KF Method for Flight Scenario 1

### 4.4.1 Results for SIFT Features and SURF Descriptors Method

Where for all **(One Kmeas)** method measurement gain value's figures, when Kmeas gain value is going closer to the value of (1) This indicates having more trust in Image processing measurement; while having its value closer to (0) implies having more trust in GNSS measurement signal in that instance of time through the UAV flight as shown in figure 4.8.

From figure 4.9 we can notice the KF algorithm dependence on the Image processing signal once it's available, Especially in scenario 2, the drone flight area included regions not covered in the training data. For that reason, the algorithm had to depend on the GNSS through its flight over unrecognized areas, by analyzing the results it can be noticed that once the live image matches with stored training data the algorithm mostly depended on the image processing signal as it provided higher accuracy input closer to the current navigator prediction in comparison to the GNSS input noisy signal.
By analyzing the graphs shown in figures 4.10, 4.11, 4.12, 4.13, and 4.14 we can notice that the biggest Error in Kalman output is showing up for all methods when a big flight maneuver changing movement direction happens, which can be seen in Scenario 1 at different times like between the time of (7s-14s) also between (27s-34s) in the previous graphs; which is caused naturally by KF algorithm which lags the estimation response to quick changes coming from the measurements input.

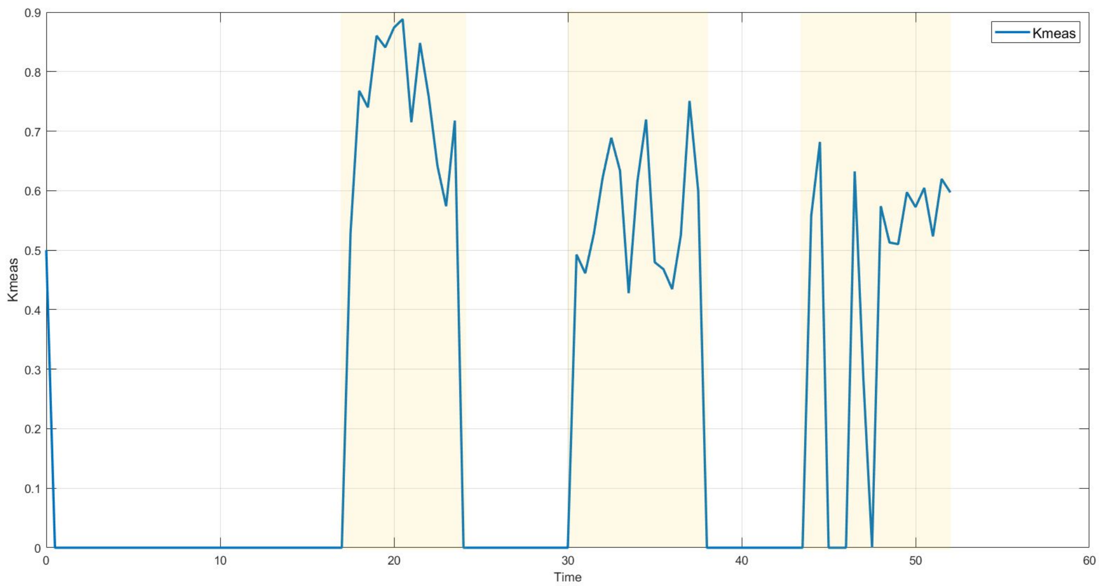


**Figure 4.9** (Kmeas) Measurement Gain values Over Time Used in 1kmeas KF Method for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

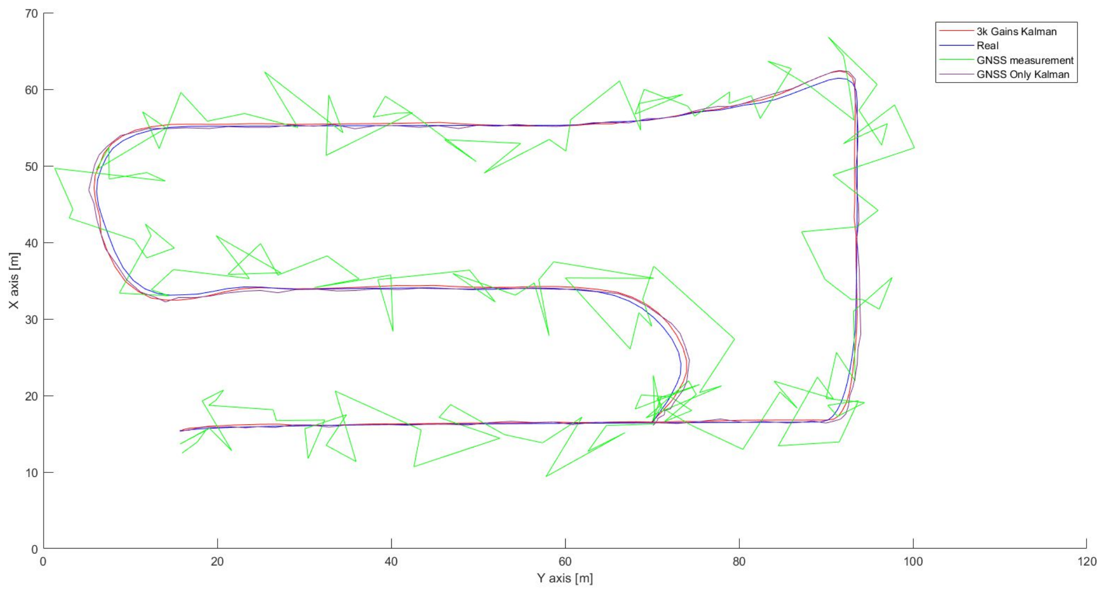


**Figure 4.11** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison with GNSS Input Signal (Green) and 3k Gains Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 1

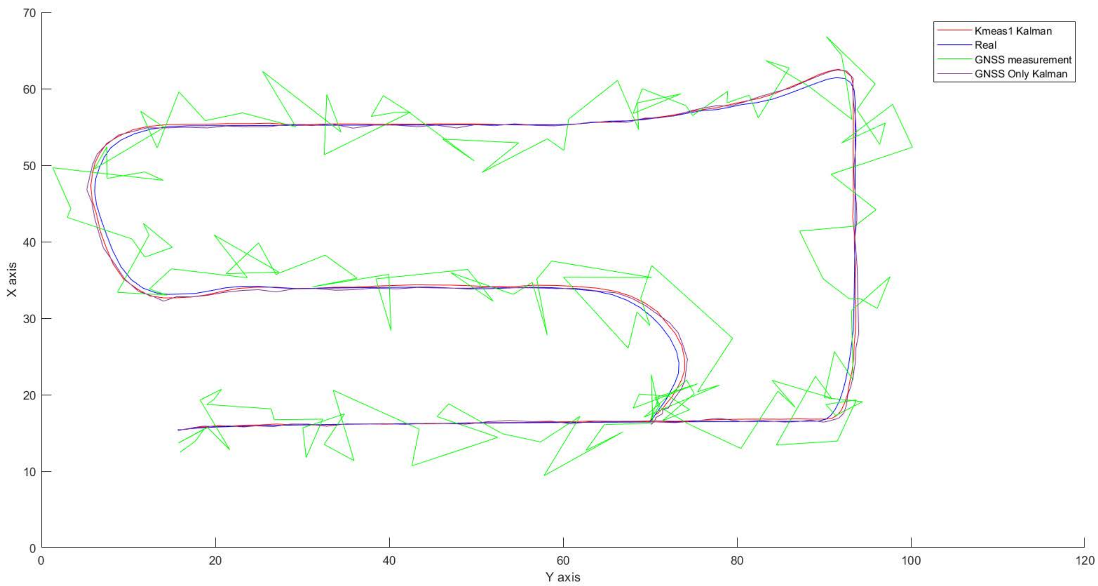


**Figure 4.10** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison with GNSS Input Signal (Green) and 1Kmeas Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 1

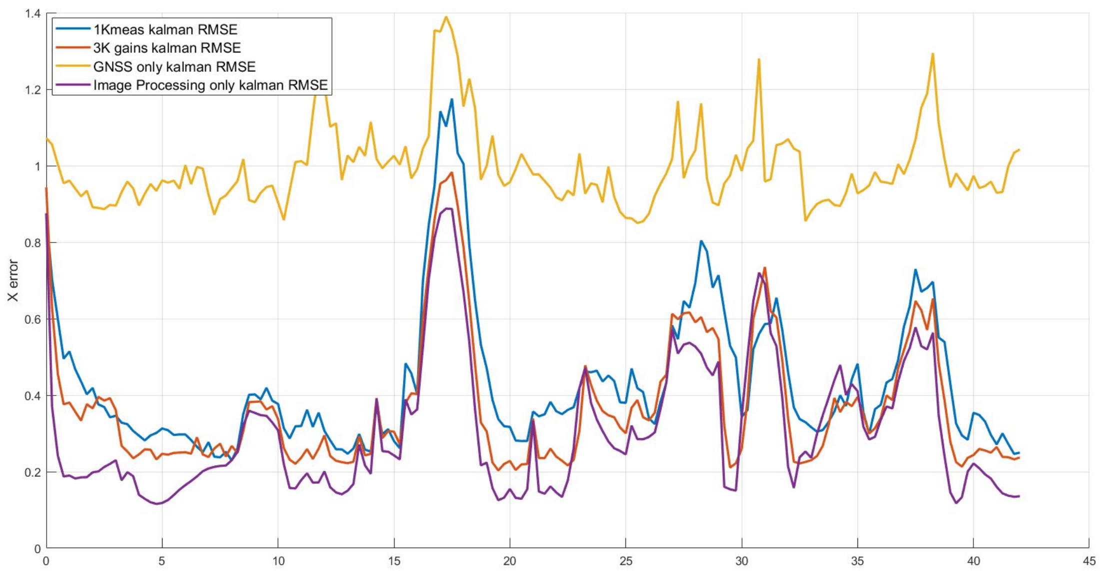


**Figure 4.12** RMSE for X Coordinate Estimations of Scenario1 Over Time

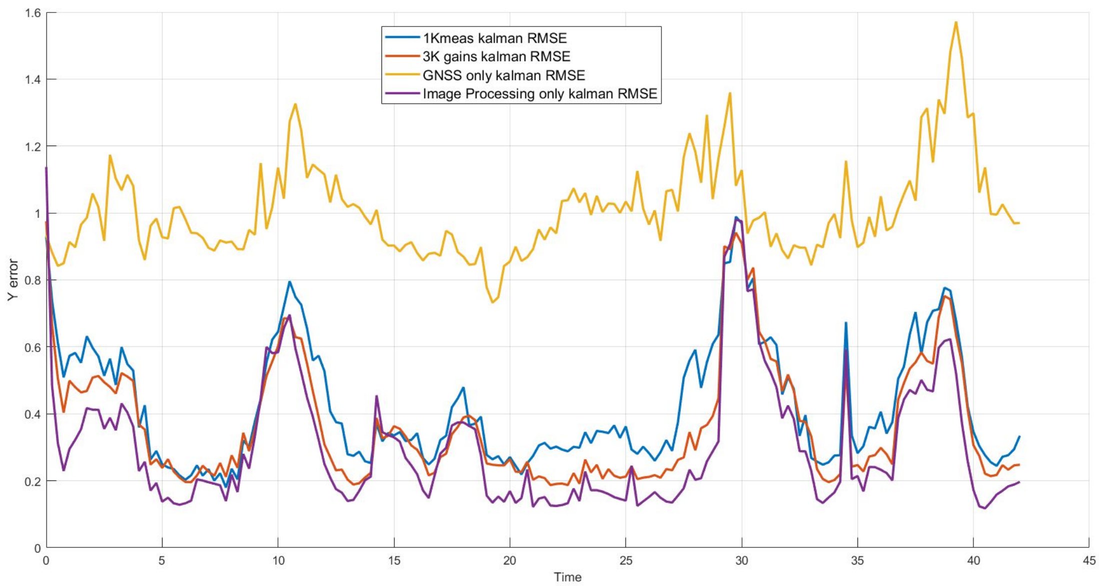


**Figure 4.13** RMSE for Y Coordinate Estimations of Scenario1 Over Time

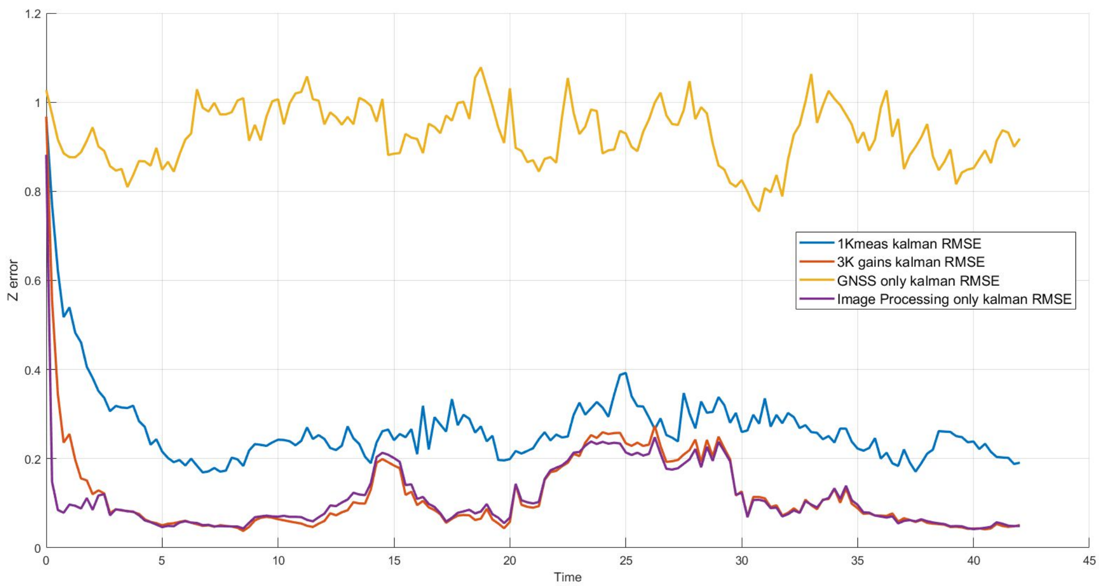


**Figure 4.14** RMSE for Z Coordinate Estimations of Scenario1 Over Time

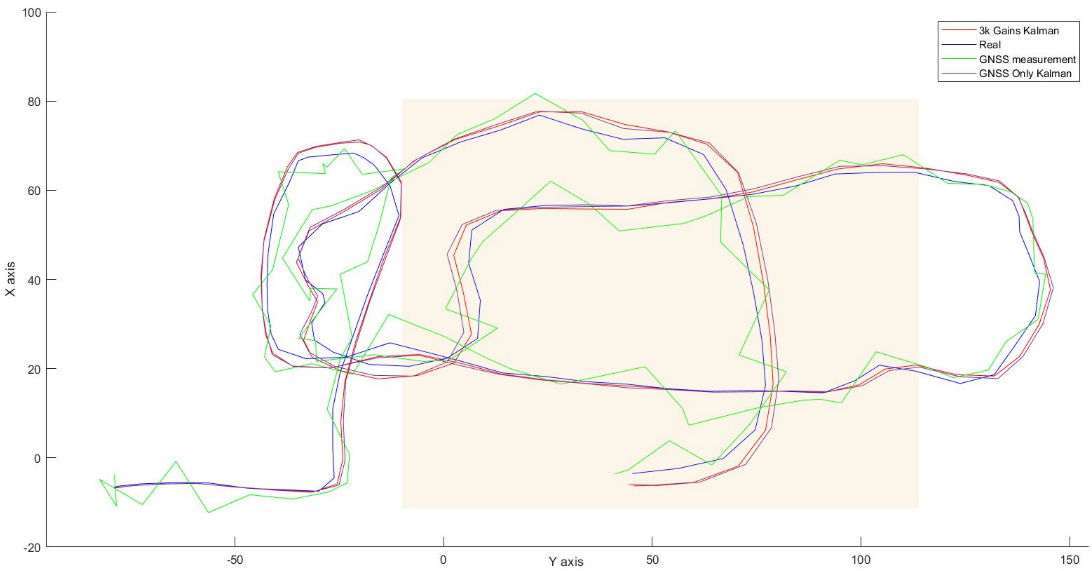


**Figure 4.15** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison With GNSS Signal Input (Green) and 3k Gains Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

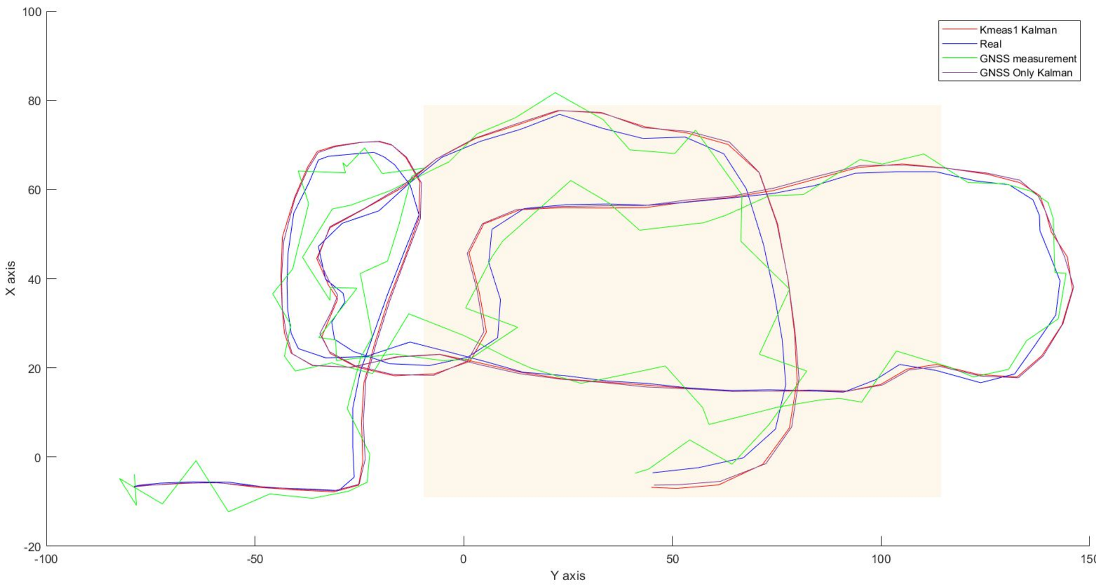


**Figure 4.16** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison With GNSS Signal Input (Green) and 1Kmeas Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

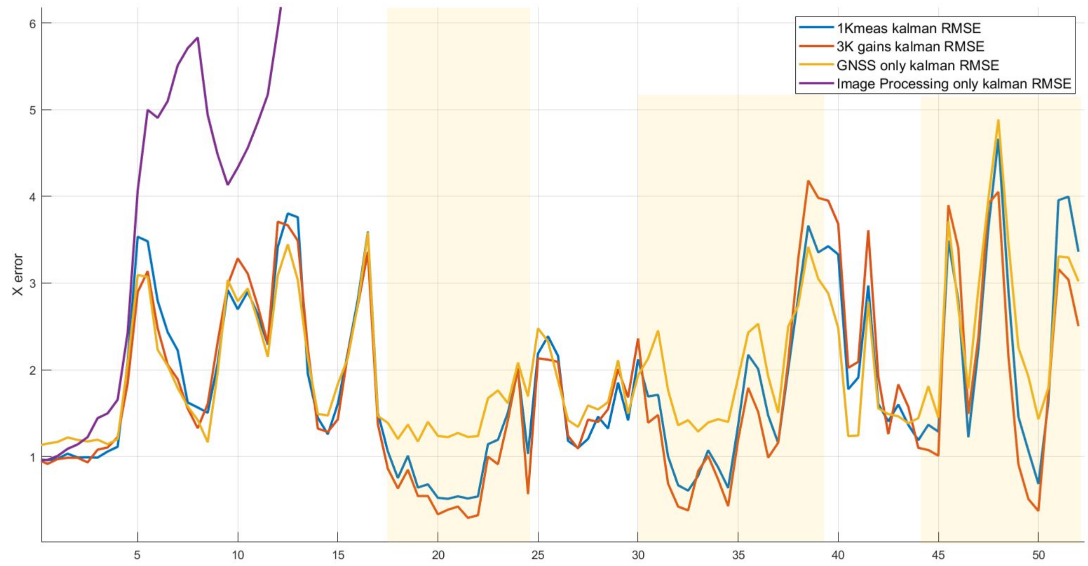


**Figure 4.17** RMSE for X Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

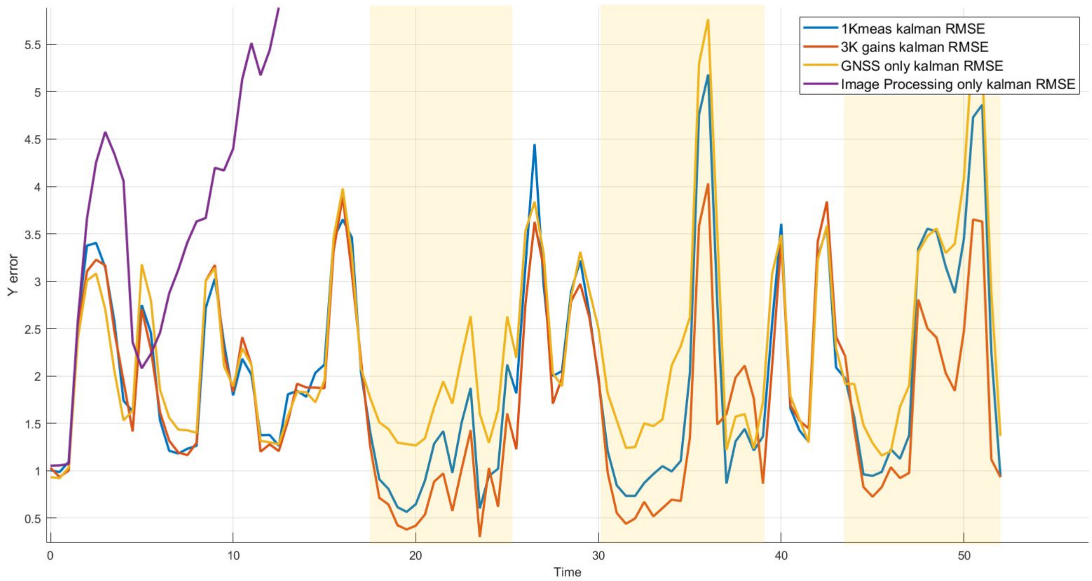


**Figure 4.18** RMSE for Z Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

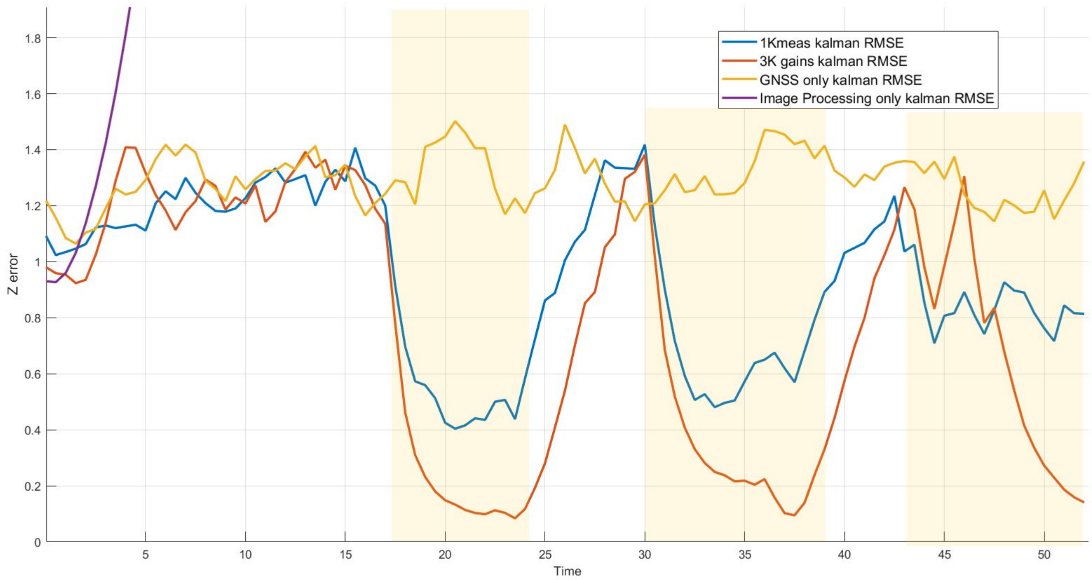


**Figure 4.19** RMSE for Y Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

In the 2nd flight scenario flying over some areas covered and some others **not covered** by the training data; caused the algorithm to have only GNSS measurement signal to depend on for some parts of the flight. In our algorithm, this will happen as far as an insufficient number of image matches were detected for **(1K and 3K)** KF methods.
In addition to that, in scenario 2 the flight path included many direction-changing maneuvers, which made it a big challenge, where in the case of only image processing being used as input measurement in KF; it can be seen obviously that the KF error accumulated over time causing a big estimation error. When there are not enough image processing matches the algorithm is only depending on the inertial navigator's predictions in the absence of an image processing signal.
To show the general dispersion of the RMSE signal of the estimations for each method and scenario separately, overall coordinates $(X, Y, Z)$ tables 4.2 and 4.3 represent numerical details of each part of these signals. Both 1Kmeas gain and 3gains Kalman filter methods included fusing GNSS, IMU, and Image Processing location estimations signals. While GNSS-only and Image processing only Kalman filter results included fusing IMU navigator signals with the main measurement signal which can be the GNSS positioning signal or image processing position estimation signal for each case.

**Table 4.2** Comparison of Output Signals' RMSE Values for All Image Processing Methods Through The 1st Flight Scenario

| 1st Flight Scenario | | | | | | | |
|---|---|---|---|---|---|---|---|
| Image Processing Method | Data TYPE | Means | | | Variances | | |
| | Signal part / Signal Type | X | Y | Z | X | Y | Z |
| SURF only features and descriptors | RMSE of 1 Kmeas gain KF | 0.4437 | 0.4560 | 0.3004 | 0.0343 | 0.0475 | 0.0072 |
| | RMSE of 3Gains KF | 0.3620 | 0.3734 | 0.1515 | 0.0286 | 0.0289 | 0.0104 |
| | RMSE of GNSS Only KF | 0.9963 | 1.0398 | 0.9431 | 0.0138 | 0.0174 | 0.0073 |
| | RMSE of Only Image Processing KF | 0.3240 | 0.3067 | 0.1327 | 0.0336 | 0.0351 | 0.0106 |
| SIFT only features and descriptors | RMSE of 1 Kmeas gain KF | 0.4836 | 0.4620 | 0.3148 | 0.0366 | 0.0309 | 0.0127 |
| | RMSE of 3Gains KF | 0.3881 | 0.4048 | 0.2581 | 0.0295 | 0.0342 | 0.0541 |
| | RMSE of GNSS Only KF | 1.0137 | 1.0028 | 0.9037 | 0.0175 | 0.0164 | 0.0048 |
| | RMSE of Only Image Processing KF | 0.3739 | 0.3334 | 0.1424 | 0.0585 | 0.0301 | 0.0146 |
| SIFT features and SURF descriptors | RMSE of 1 Kmeas gain KF | 0.4334 | 0.4232 | 0.2703 | 0.0347 | 0.0330 | 0.0089 |
| | RMSE of 3Gains KF | 0.3716 | 0.3662 | 0.1197 | 0.0295 | 0.0323 | 0.0102 |
| | RMSE of GNSS Only KF | 0.9942 | 1.0085 | 0.9287 | 0.0107 | 0.0194 | 0.0044 |
| | RMSE of Only Image Processing KF | 0.3084 | 0.3022 | 0.1129 | 0.0314 | 0.0368 | 0.0071 |
| SURF features and SIFT descriptors | RMSE of 1 Kmeas gain KF | 0.4450 | 0.4216 | 0.2856 | 0.0358 | 0.0286 | 0.0071 |
| | RMSE of 3Gains KF | 0.3596 | 0.3780 | 0.1171 | 0.0314 | 0.0334 | 0.0087 |
| | RMSE of GNSS Only KF | 0.9942 | 1.0085 | 0.9287 | 0.0107 | 0.0194 | 0.0044 |
| | RMSE of Only Image Processing KF | 0.3117 | 0.3013 | 0.1147 | 0.0312 | 0.0335 | 0.0087 |

**Table 4.3** Comparison of Output Signals' RMSE Values for All Image Processing Methods Through The 2nd Flight Scenario

| 2nd Flight Scenario | | | | | | | |
|---|---|---|---|---|---|---|---|
| Image Processing Method | Data TYPE | Means | | | Variances | | |
| | Signal part / Signal Type | X | Y | Z | X | Y | Z |
| SURF only features and descriptors | RMSE of 1 Kmeas gain KF | 1.8140 | 2.0029 | 0.9403 | 0.8982 | 1.1933 | 0.1107 |
| | RMSE of 3Gains KF | 1.7303 | 1.8039 | 0.7097 | 1.0515 | 1.0961 | 0.2068 |
| | RMSE of GNSS Only KF | 2.0493 | 2.2340 | 1.2882 | 0.6298 | 1.0234 | 0.0165 |
| | RMSE of Only Image Processing KF | 21.2316 | 19.2782 | 22.6308 | 203.1080 | 148.8846 | 272.7566 |
| SIFT only features and descriptors | RMSE of 1 Kmeas gain KF | 1.8701 | 1.9937 | 1.0207 | 0.8509 | 0.9982 | 0.0917 |
| | RMSE of 3Gains KF | 1.8303 | 1.8022 | 0.7976 | 1.0083 | 0.8057 | 0.1892 |
| | RMSE of GNSS Only KF | 2.0493 | 2.2340 | 1.2882 | 0.6298 | 1.0234 | 0.0165 |
| | RMSE of Only Image Processing KF | 19.9437 | 20.8090 | 20.1315 | 197.3783 | 209.6566 | 210.1162 |
| SIFT features and SURF descriptors | RMSE of 1 Kmeas gain KF | 1.8531 | 1.9997 | 0.9423 | 0.9947 | 1.1720 | 0.0858 |
| | RMSE of 3Gains KF | 1.7887 | 1.7776 | 0.7600 | 1.1127 | 0.9631 | 0.2128 |
| | RMSE of GNSS Only KF | 2.0047 | 2.2574 | 1.2920 | 0.6163 | 1.0644 | 0.0090 |
| | RMSE of Only Image Processing KF | 20.6411 | 20.1694 | 24.2763 | 231.1866 | 208.8980 | 309.6648 |
| SURF features and SIFT descriptors | RMSE of 1 Kmeas gain KF | 1.8291 | 1.9811 | 0.9332 | 0.8567 | 1.0727 | 0.0929 |
| | RMSE of 3Gains KF | 1.6975 | 1.7435 | 0.7407 | 0.9356 | 0.9892 | 0.2139 |
| | RMSE of GNSS Only KF | 2.0047 | 2.2574 | 1.2920 | 0.6163 | 1.0644 | 0.0090 |
| | RMSE of Only Image Processing KF | 18.4899 | 20.1971 | 19.6219 | 145.9892 | 177.0660 | 203.9881 |

### 4.4.2 Results for SURF Only Features and Descriptors Method

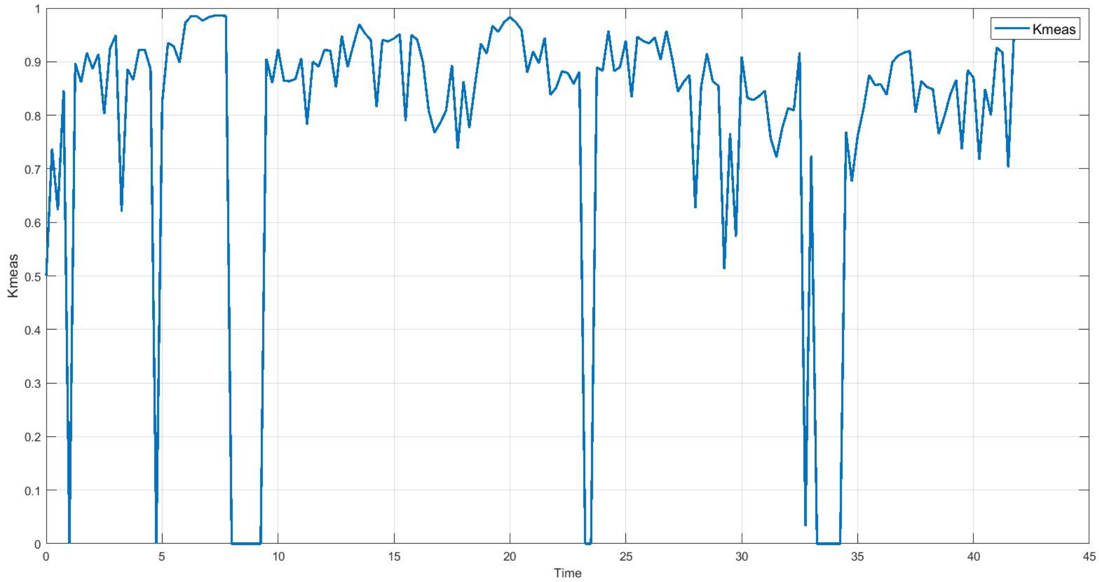


**Figure 4.20** (One Kmeas) Measurement Gain Values Over Time Used in 1kmeas KF Method for Flight Scenario 1

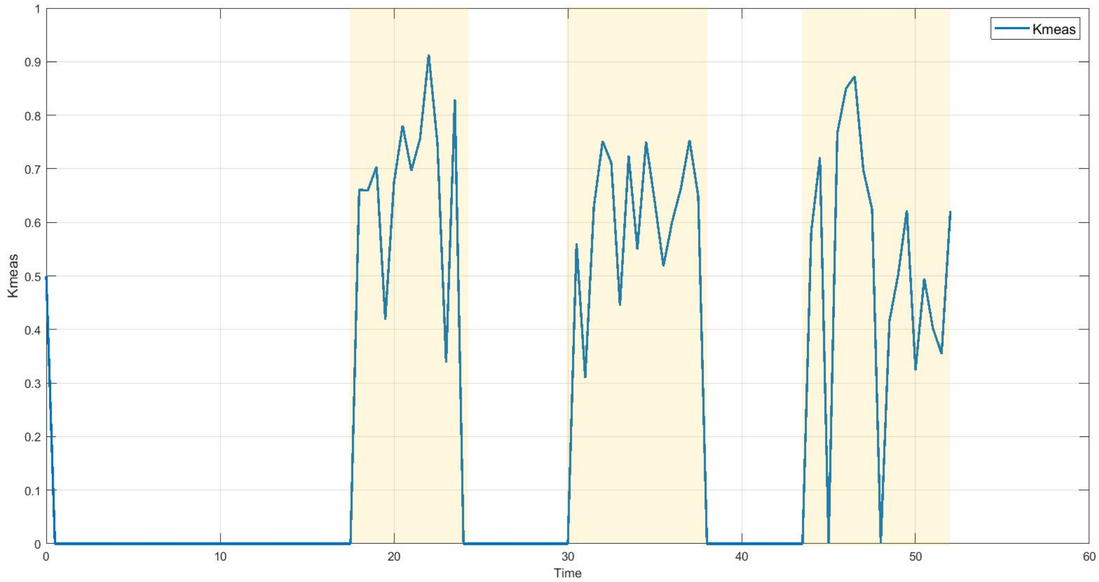


**Figure 4.21** (1 Kmeas) Measurement Gain Values Over Time Used in 1kmeas KF Method for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

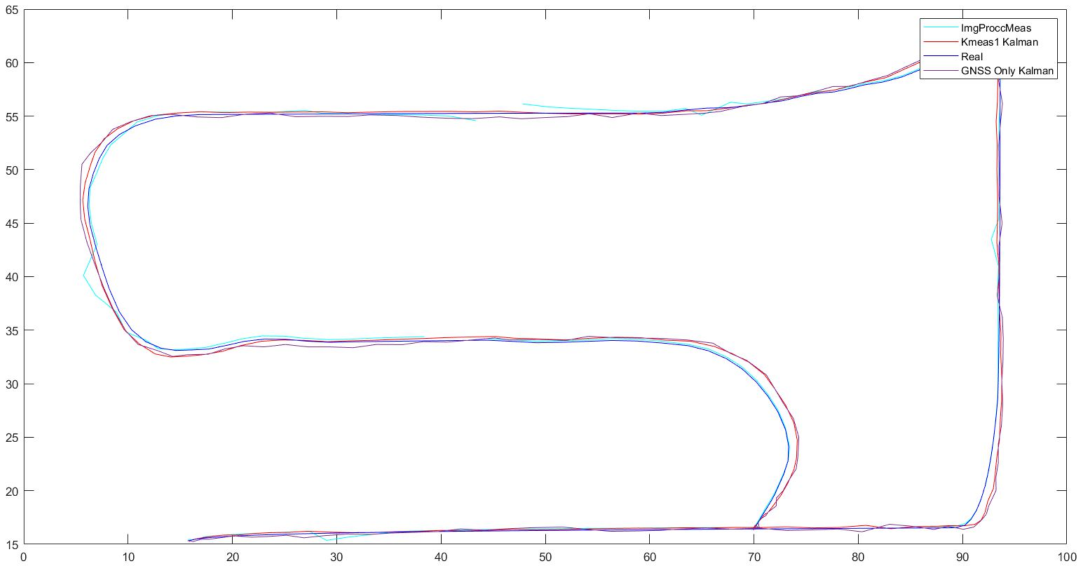


**Figure 4.22** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison with Image Processing Signal Input (Cyan) and 1Kmeas Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 1

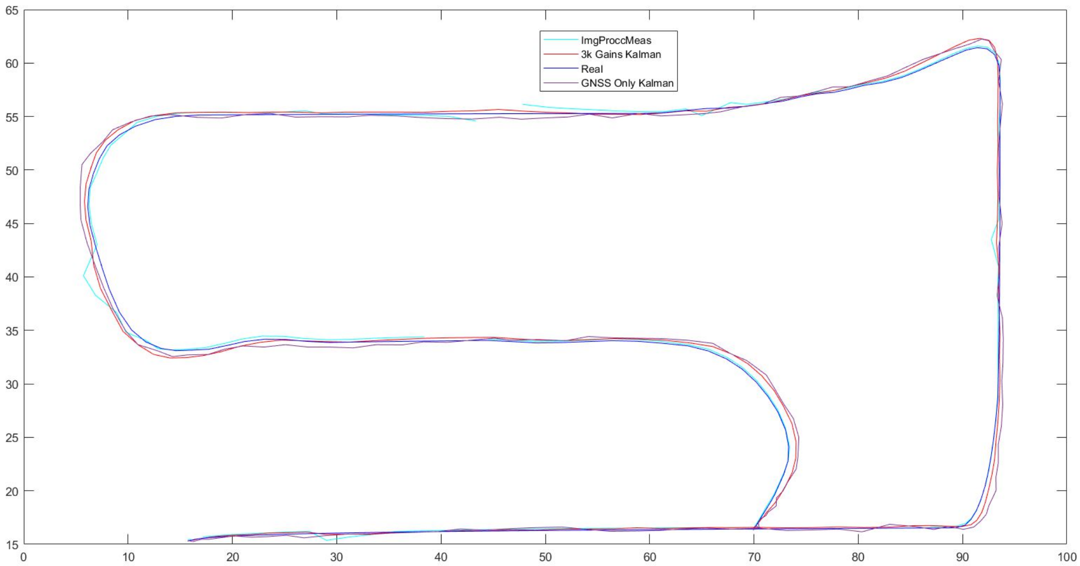


**Figure 4.23** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison With Image Processing Signal Input (Cyan) and 3k Gains Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 1

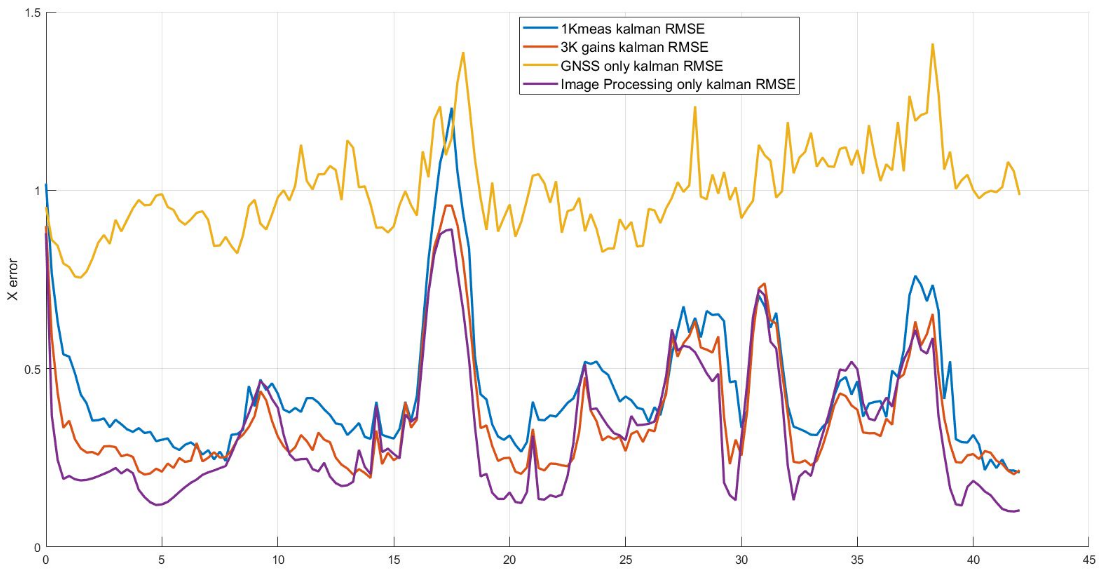


**Figure 4.24** RMSE for Y Coordinate Estimations of Scenario1 Over Time

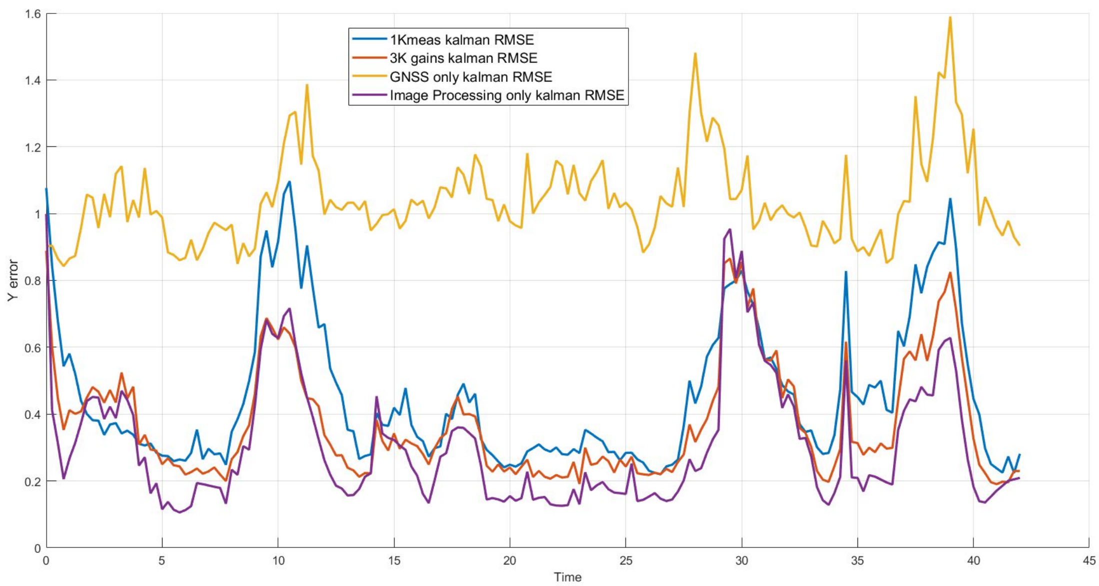


**Figure 4.25** RMSE for X Coordinate Estimations of Scenario1 Over Time


**Figure 4.26** RMSE for Z Coordinate Estimations of Scenario1 Over Time

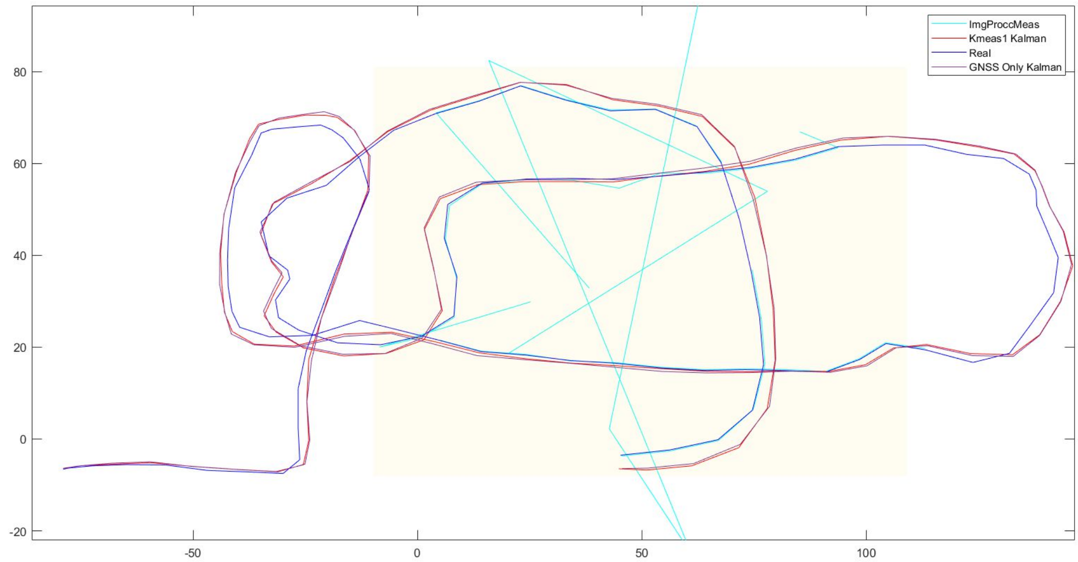


**Figure 4.27** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison With Image Processing Signal Input (Cyan) and 1Kmeas Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

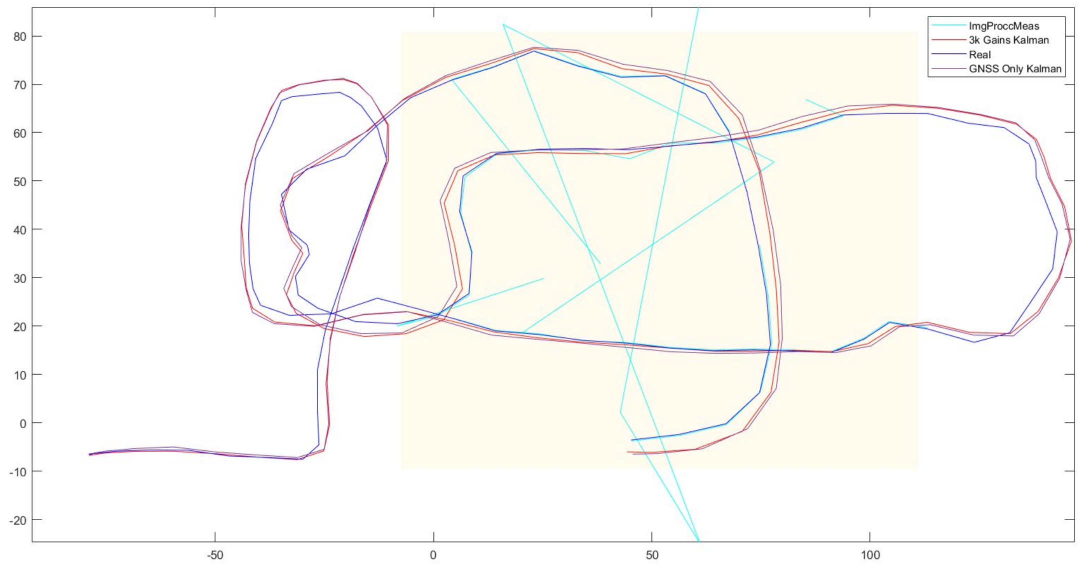


**Figure 4.28** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison with Image Processing Signal Input (Cyan) and 3k Gains Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

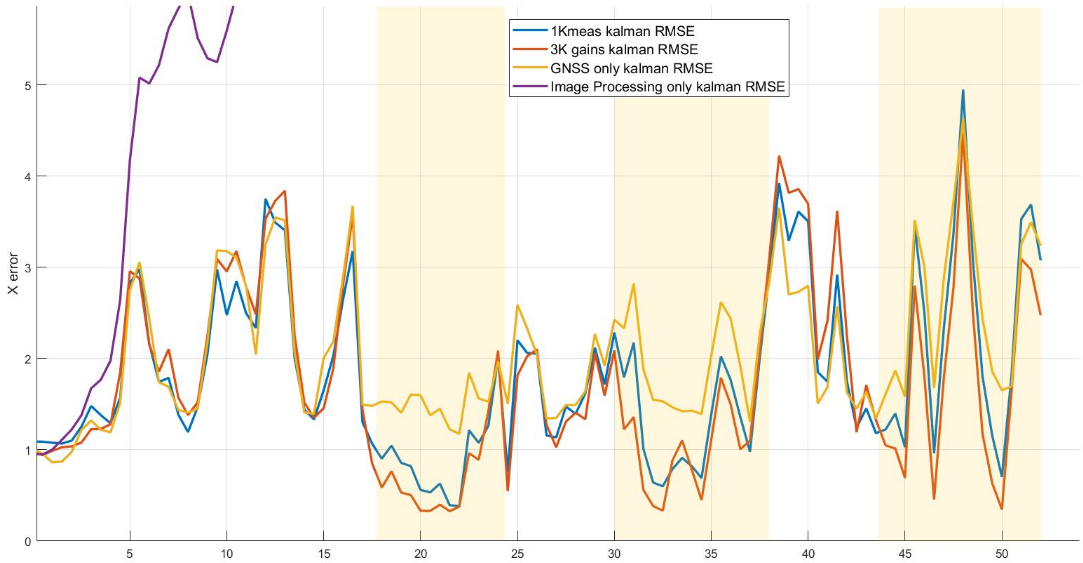


**Figure 4.29** RMSE for X Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

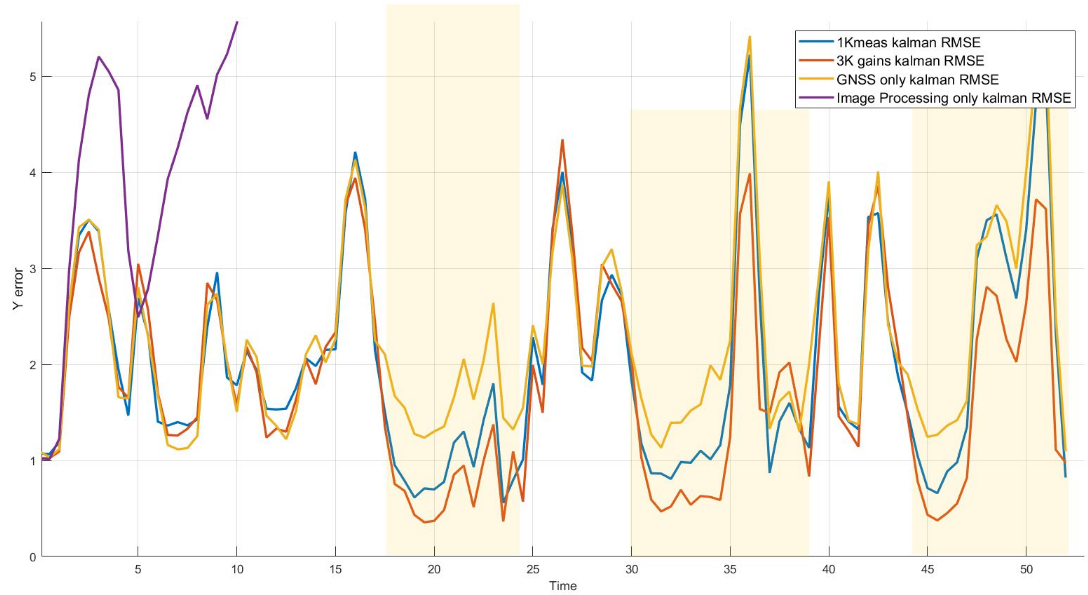


**Figure 4.30** RMSE for Y Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

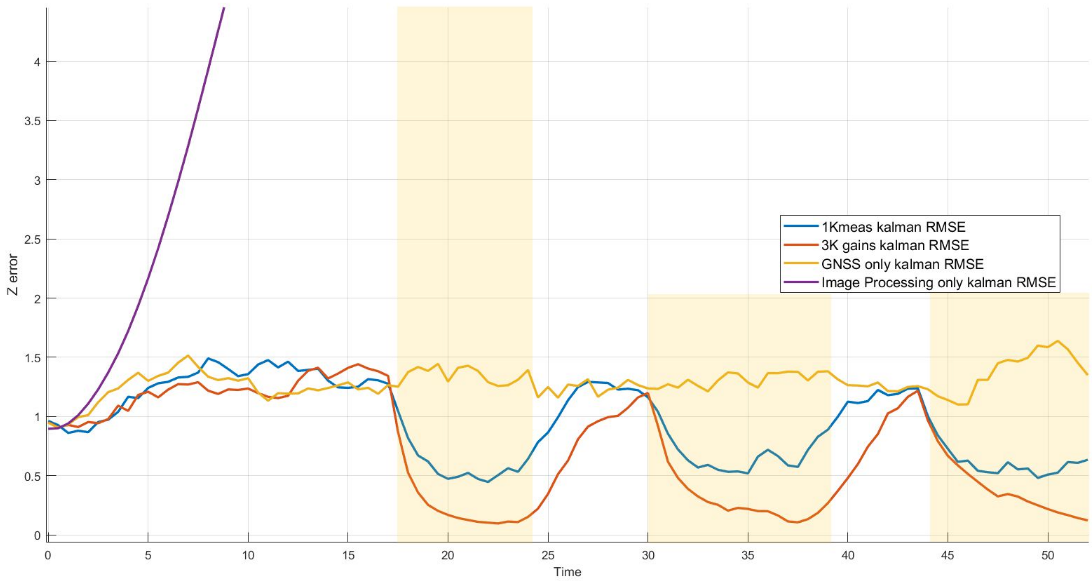


**Figure 4.31** RMSE for Z Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

### 4.4.3 Results for SURF Features and SIFT Descriptors Method

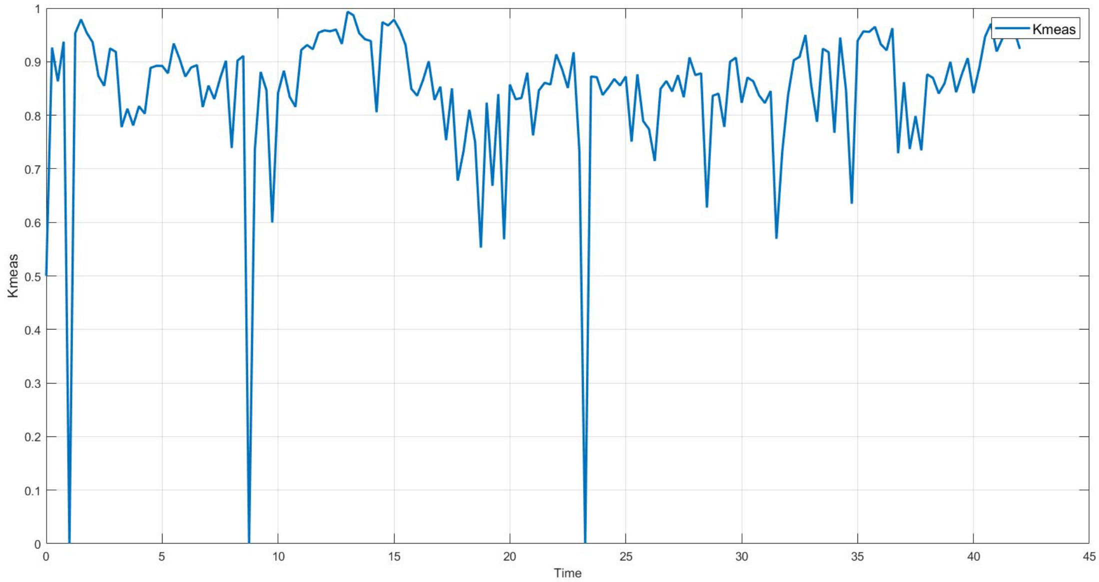


**Figure 4.32** (Kmeas) Measurement Gain Values Over Time Used in 1kmeas KF Method for Flight Scenario 1

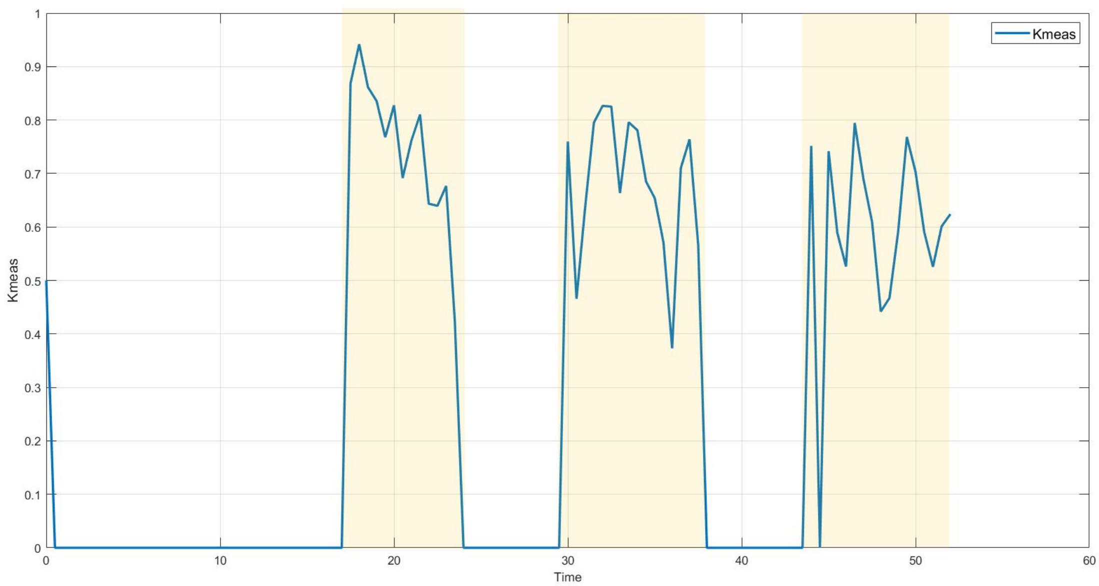


**Figure 4.33** (Kmeas) Measurement Gain Values Over Time Used in 1kmeas KF Method for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

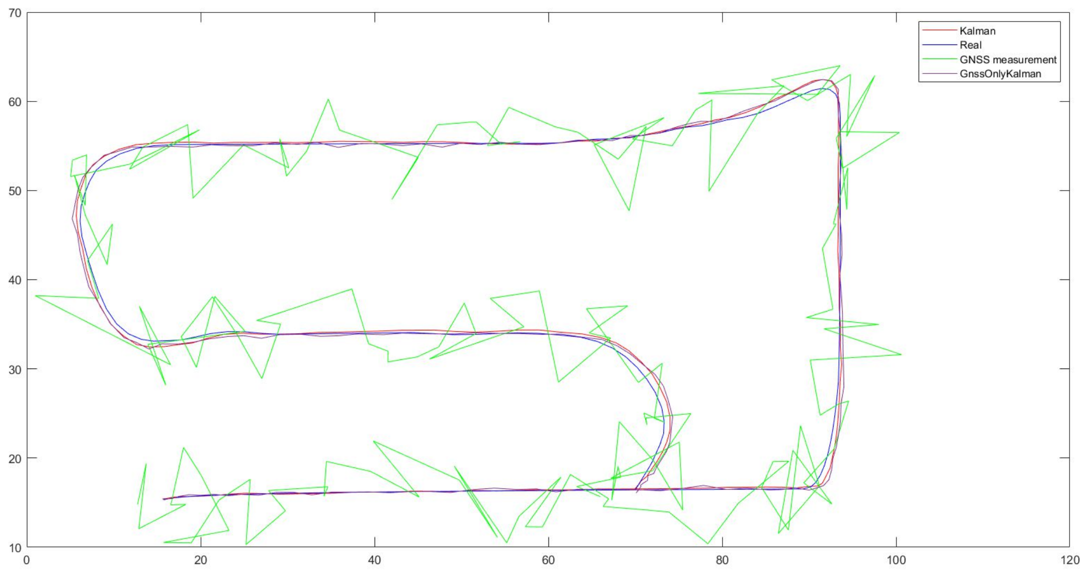


**Figure 4.34** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison With GNSS Input Signal (Green) and 1Kmeas Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 1

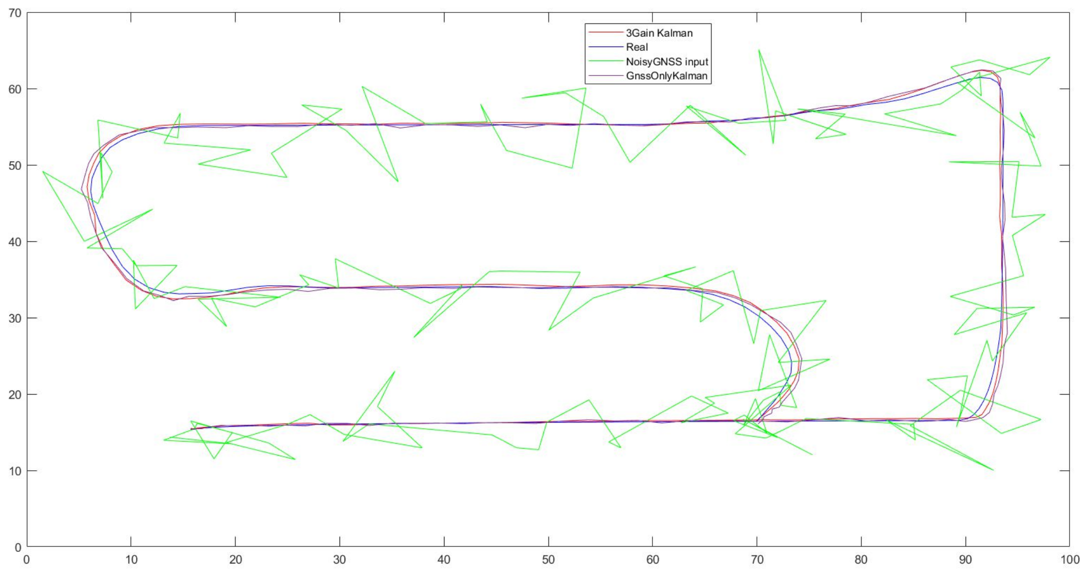


**Figure 4.35** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison With GNSS Input Signal (Green) and 3k Gains Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 1

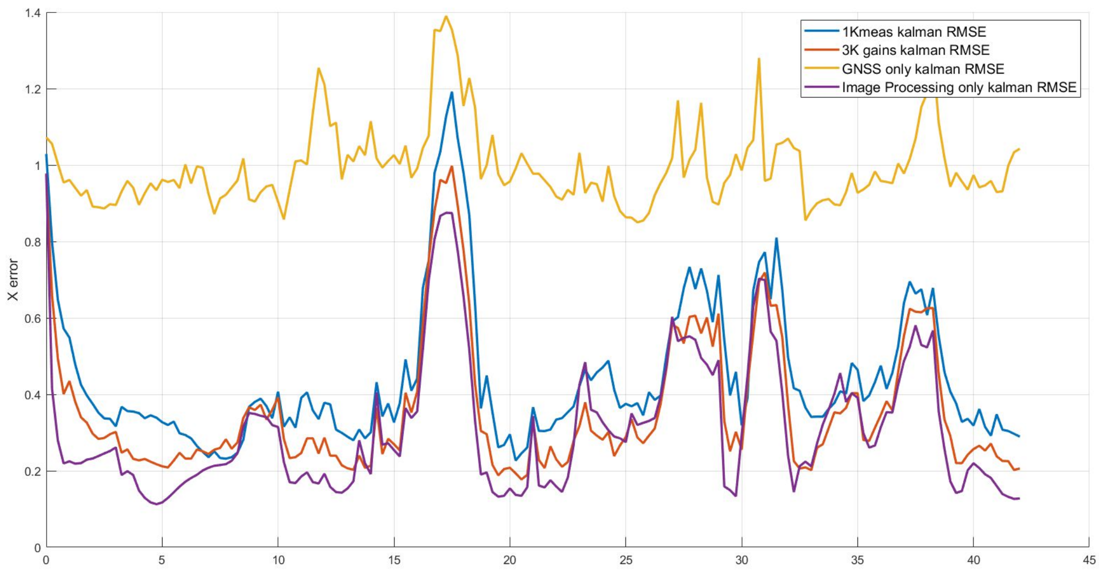


**Figure 4.36** RMSE for X Coordinate Estimations of Scenario1 Over Time

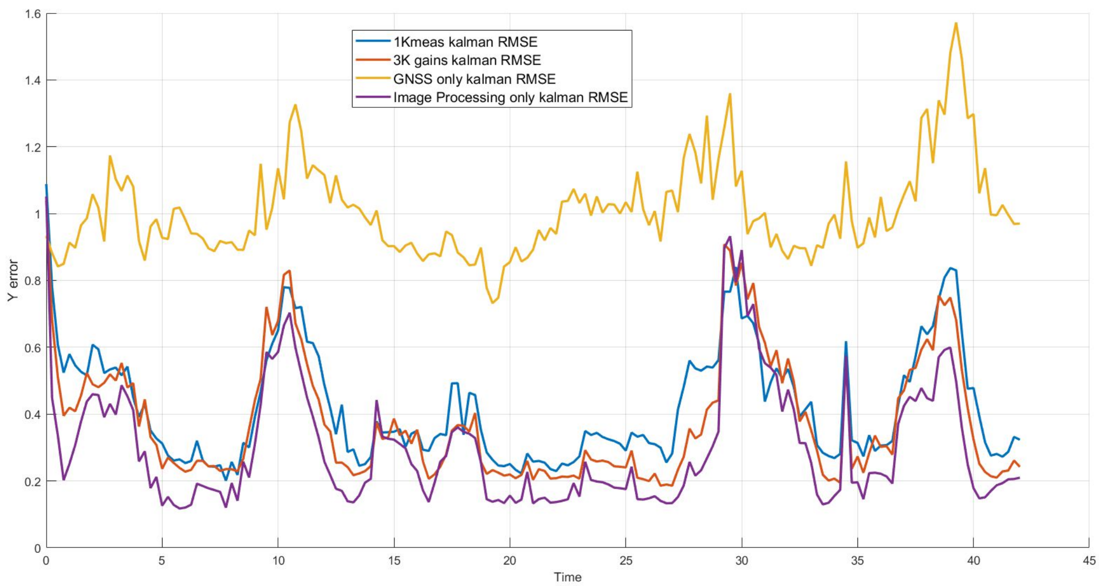


**Figure 4.37** RMSE for Y Coordinate Estimations of Scenario1 Over Time

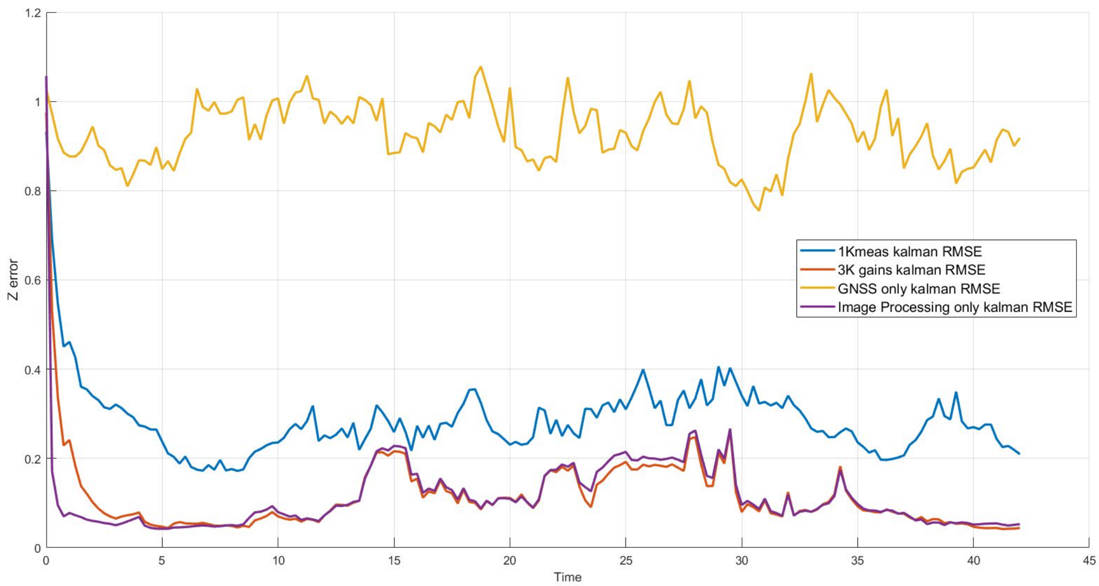


**Figure 4.38** RMSE for Z Coordinate Estimations of Scenario1 Over Time


**Figure 4.39** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison With GNSS Input Signal (Green) and 1Kmeas Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

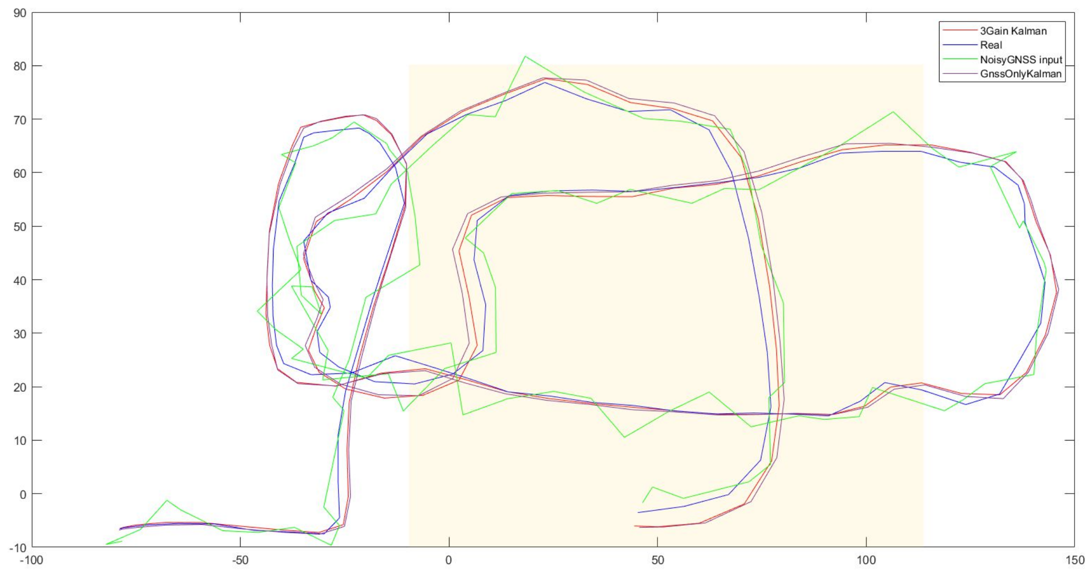


**Figure 4.40** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison With GNSS Input Signal (Green) and 3k Gains Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

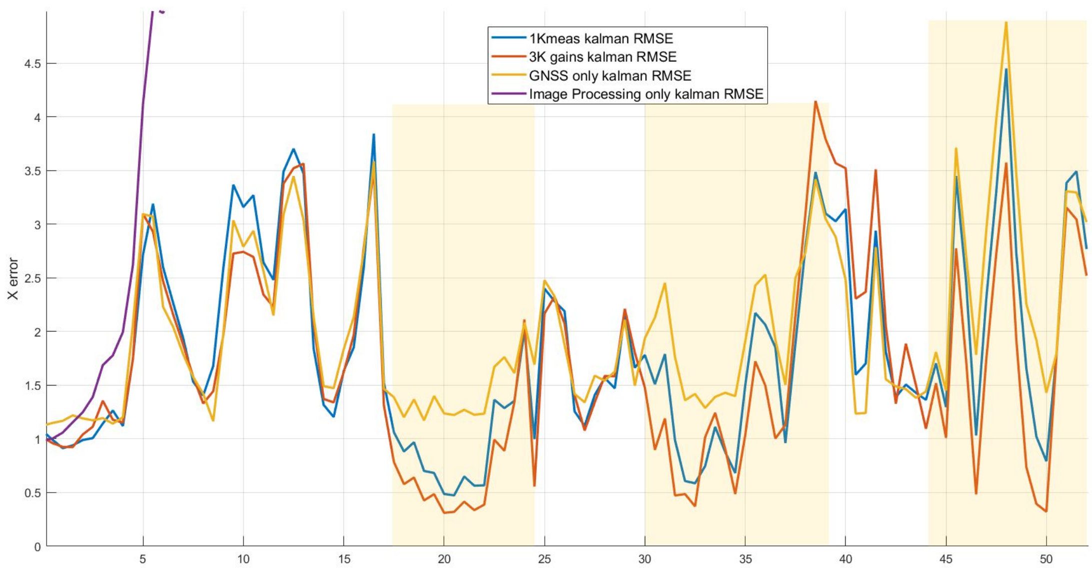


**Figure 4.41** RMSE for X Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

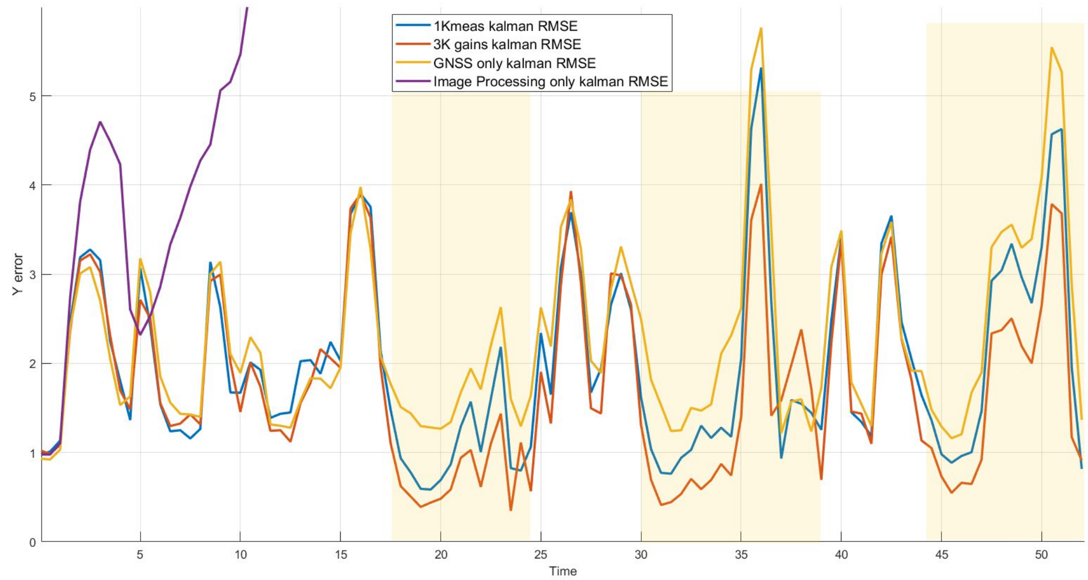


**Figure 4.42** RMSE for Y Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

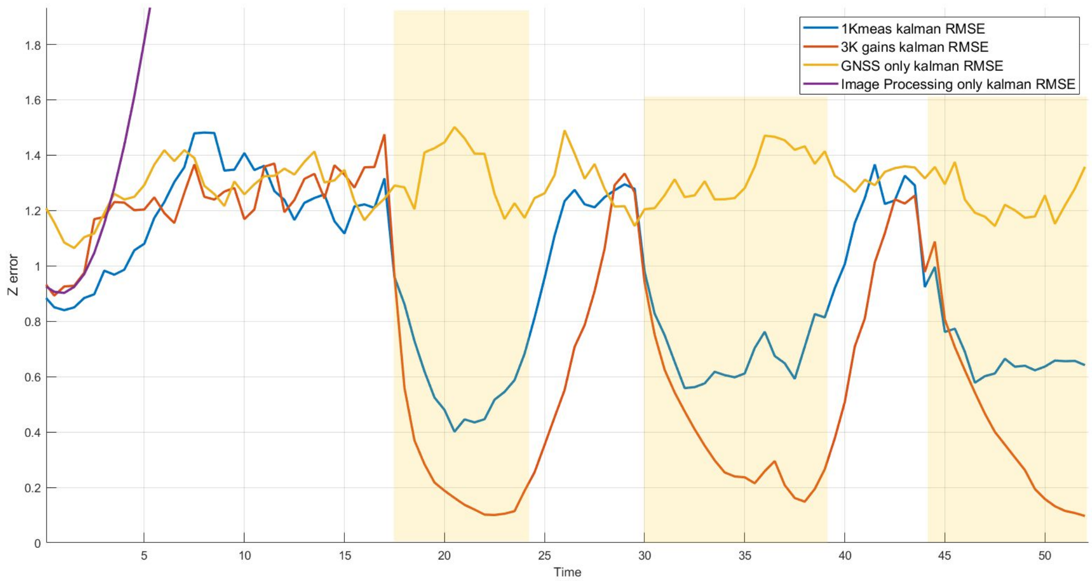


**Figure 4.43** RMSE for Z Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

### 4.4.4 Results for SIFT Only Features and Descriptors Method

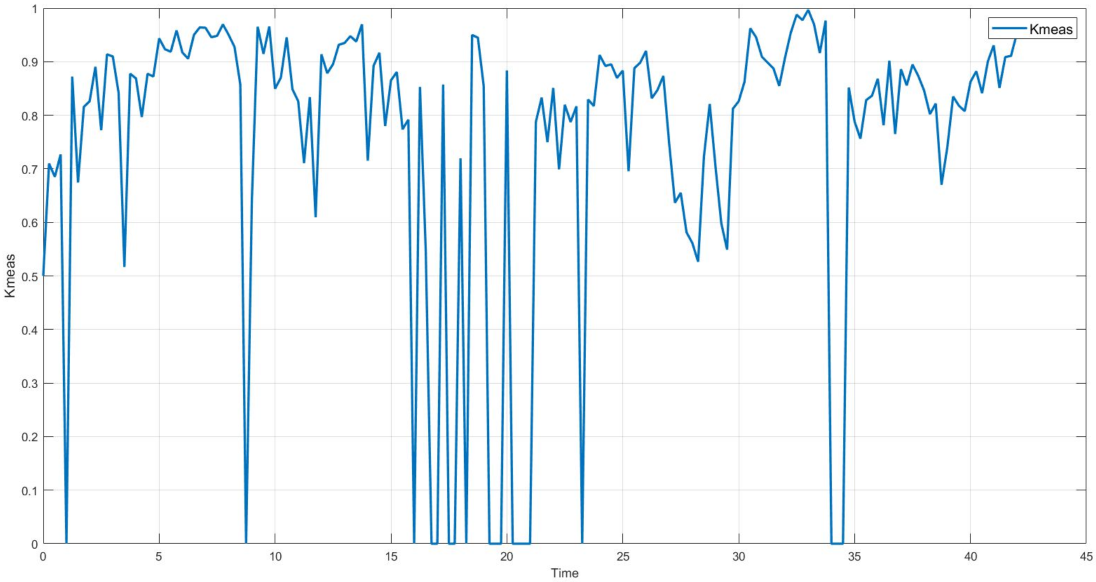


**Figure 4.44** (Kmeas) Measurement Gain values Over Time Used in 1kmeas KF Method for Flight Scenario 1

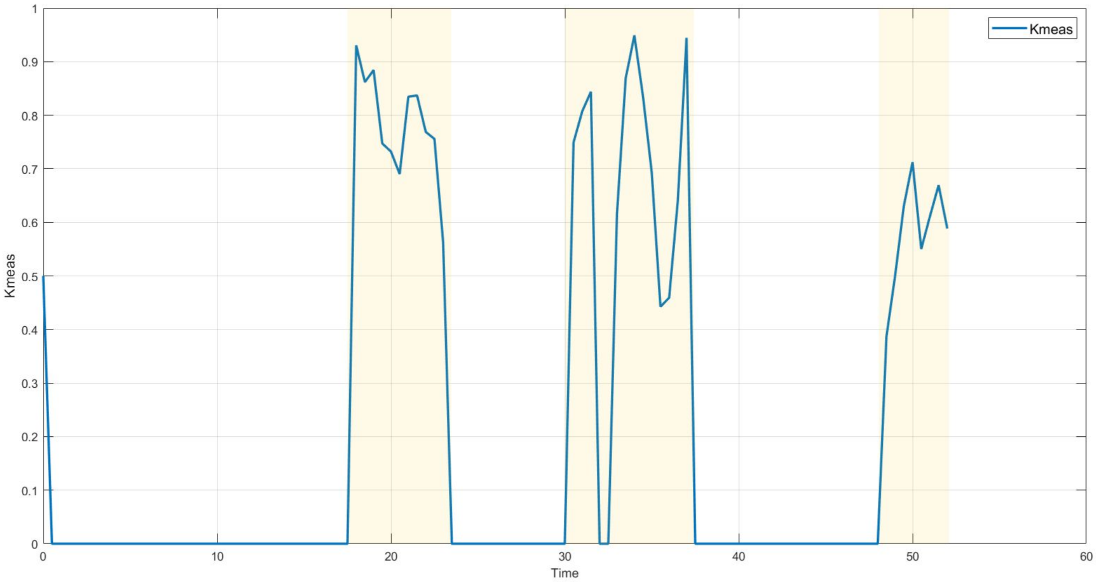


**Figure 4.45** (Kmeas) Measurement Gain Values Over Time Used in 1kmeas KF Method for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

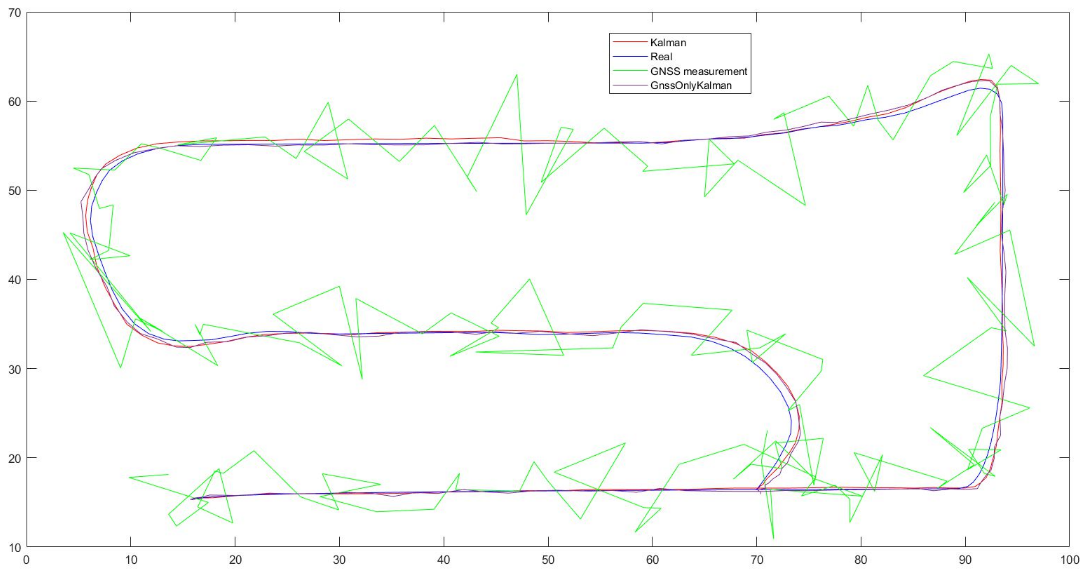


**Figure 4.46** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison With GNSS Input Signal (Green) and 1Kmeas Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 1

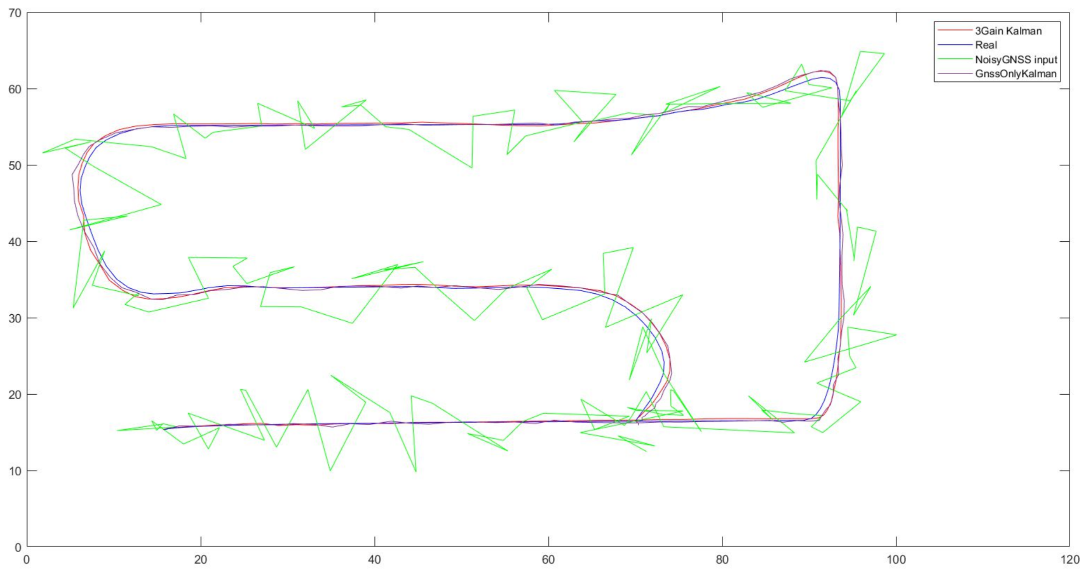


**Figure 4.47** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison with GNSS Input Signal (Green) and 3kGains Method Kalman output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 1

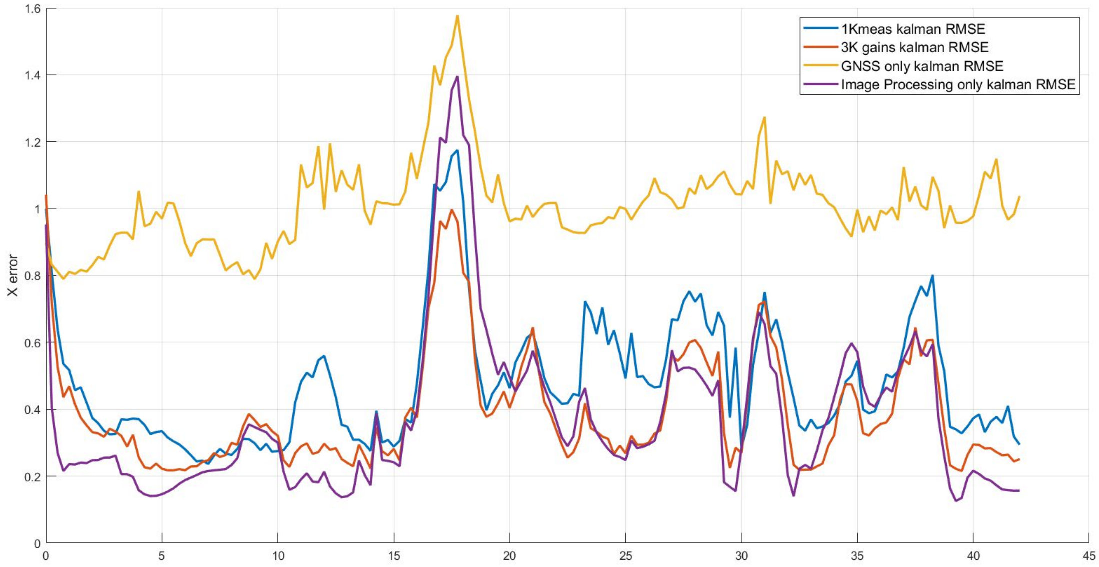


**Figure 4.48** RMSE for X Coordinate Estimations of Scenario1 Over Time

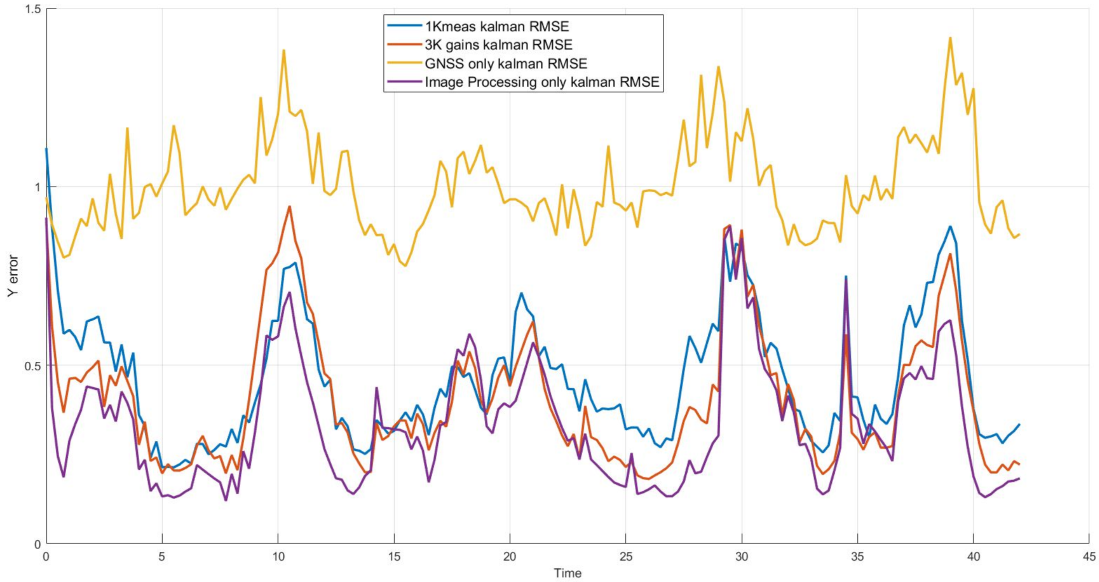


**Figure 4.49** RMSE for Y Coordinate Estimations of Scenario1 Over Time

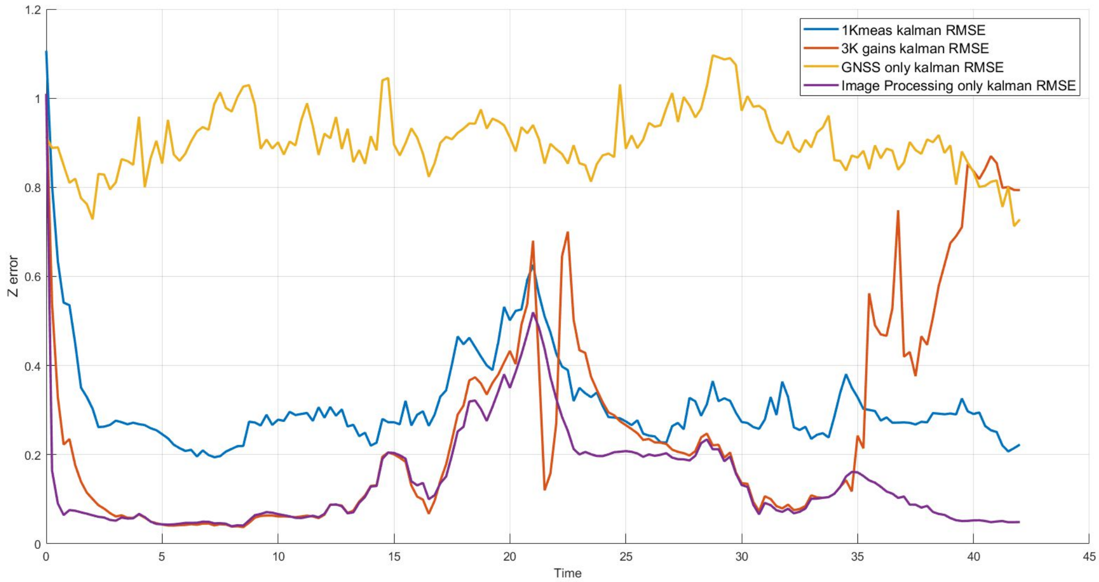


**Figure 4.50** RMSE for Z Coordinate Estimations of Scenario1 Over Time

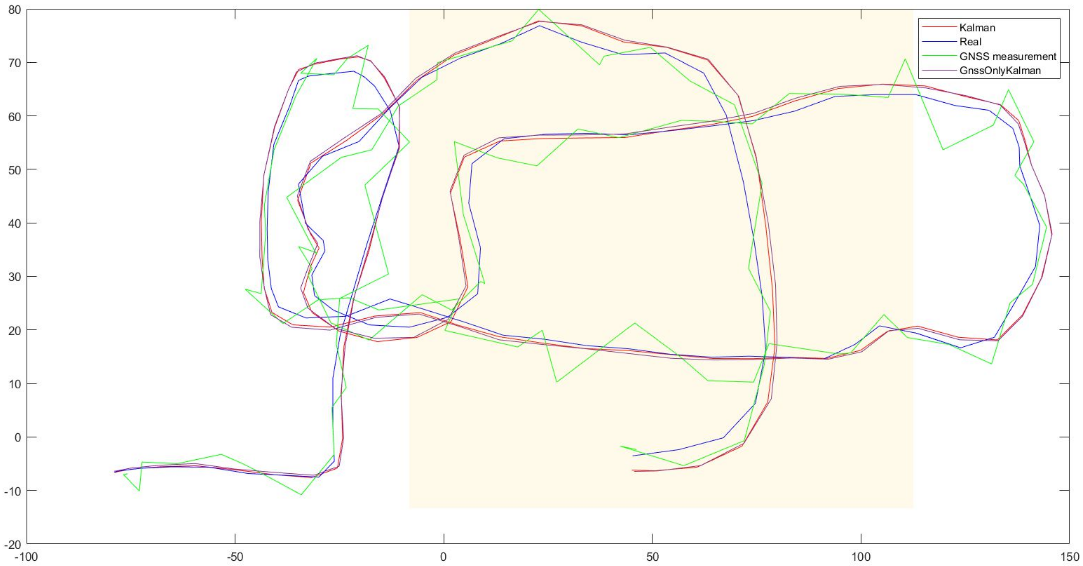


**Figure 4.51** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison with GNSS Input Signal (Green) and 1Kmeas Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

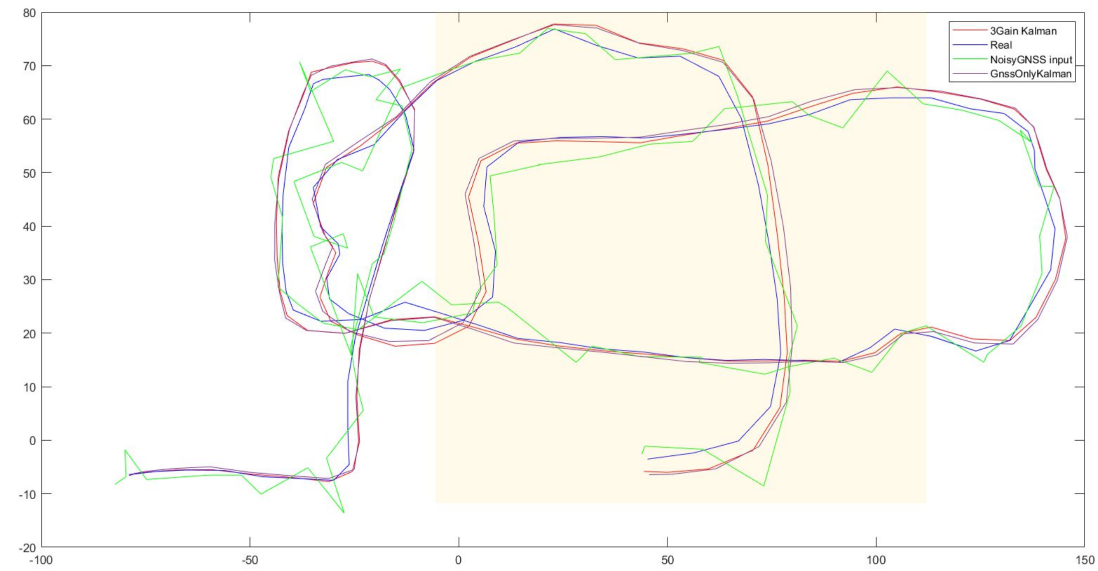


**Figure 4.52** Vertical View of Flight Path and Estimation Results Showing: Real Flight Data(Blue) in Comparison With GNSS Input Signal (Green) and 3k Gains Method Kalman Output(Red) and GNSS Only Kalman Output(Purple) for Flight Scenario 2 (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

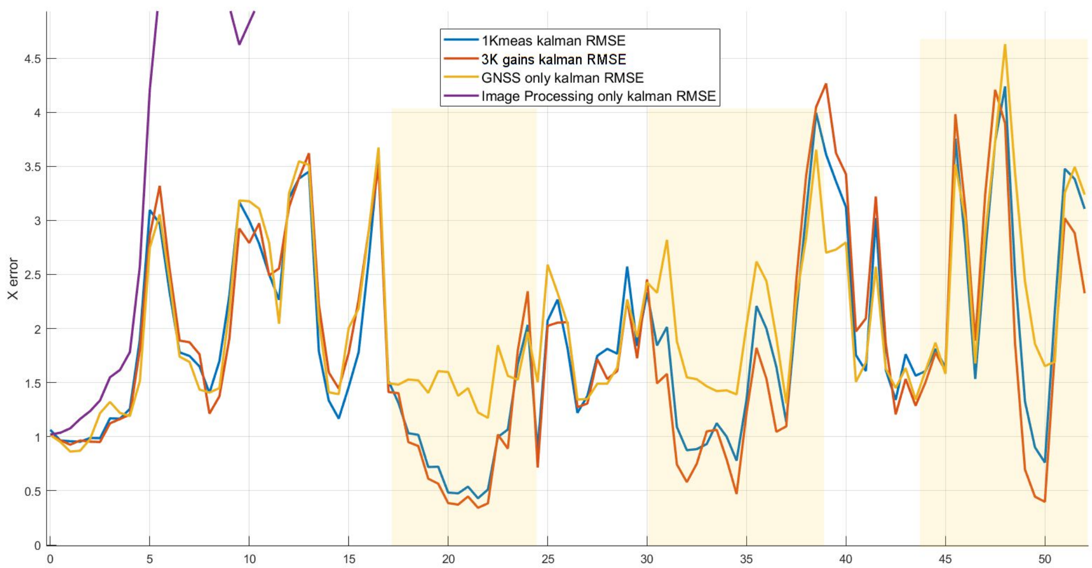


**Figure 4.53** RMSE for X Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

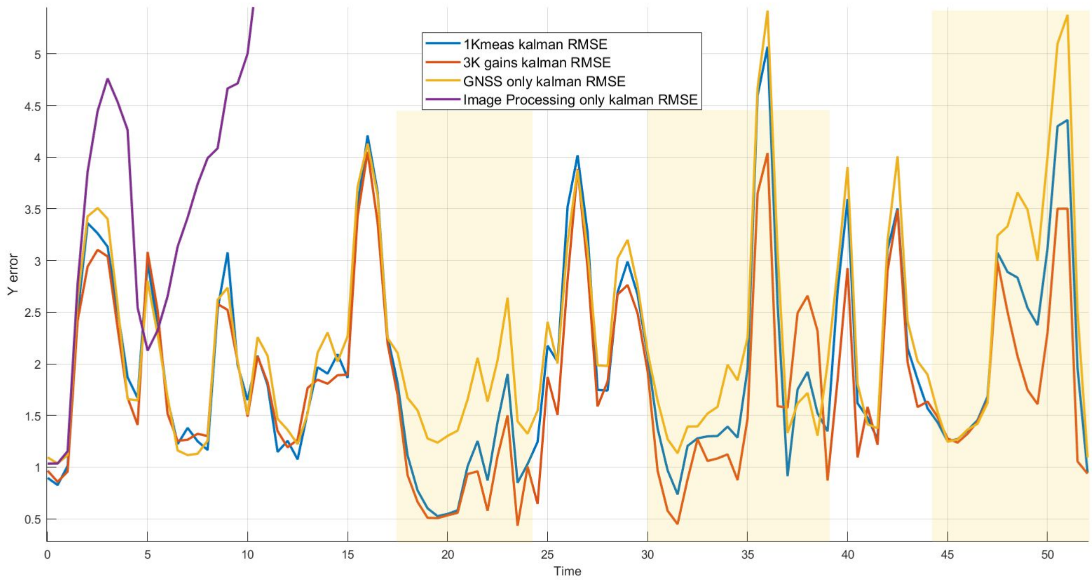


**Figure 4.54** RMSE for Y Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

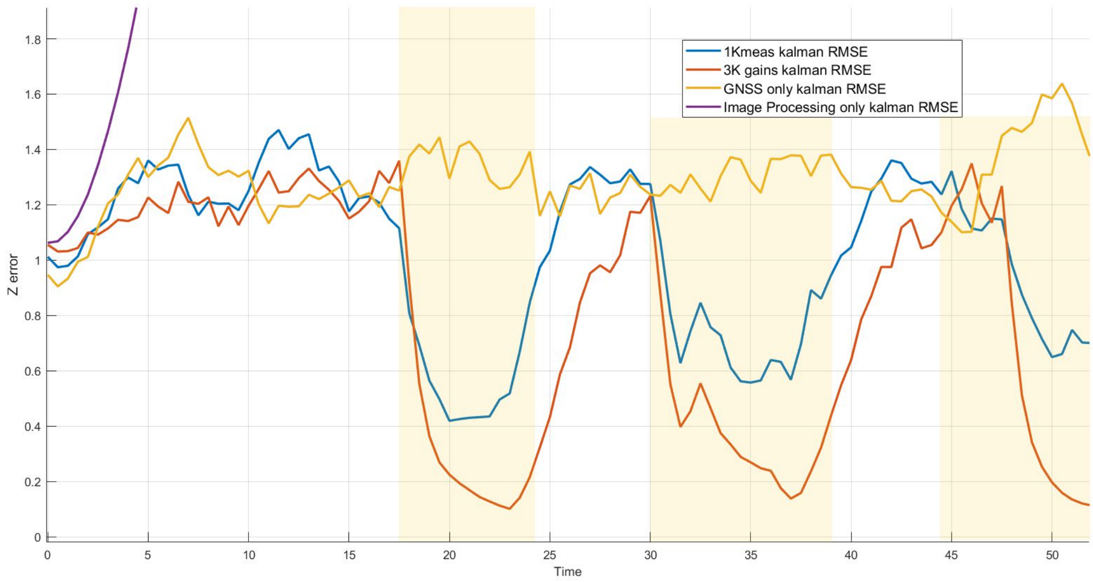


**Figure 4.55** RMSE for Z Coordinate Estimations of Scenario2 Over Time (Orange Area Refers to Passing Over Regions Included in Training Data Panorama Images)

# 5
# RESULTS AND DISCUSSION

This thesis presents a supportive vision-based navigation algorithm, where a unity engine simulation-based two aerial image mosaics of two different altitudes were created using two recorded training flight data, then two test data flight simulations were simulated and recorded. The proposed algorithm is combining four scale, and rotation invariant image processing methods, and two multi-sensor fusion Kalman filter algorithms. The experimental results showed that the proposed algorithms are reliable and accurate, where the combination of the detection and extraction of SIFT features and SURF descriptors with 3K gain KF sensor fusion method has shown the best results in terms of having more extracted features and mostly having less mean and variation values of RMSE signal in comparison with other methods combinations, which is suitable for use in the autonomous navigation of UAVs.

# REFERENCES


[1] L. Forlenza, G. Fasano, D. Accardo, A. Moccia, "Flight performance analysis of an image processing algorithm for integrated sense-and-avoid systems," *International Journal of Aerospace Engineering*, vol. 2012, Jun. 2012. DOI: 10.1155/2012/542165.

[2] *Avionics Navigation Systems*. John Wiley & Sons, Ltd, 1997, vol. 2, ISBN: 9780470172704. DOI: https://doi.org/10.1002/9780470172704.fmatter. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470172704.fmatter. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470172704.fmatter.

[3] H. Bay, T. Tuytelaars, L. Van Gool, "Surf: Speeded up robust features.," vol. 110, Jan. 2006, pp. 404–417.

[4] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov. 2004. DOI: 10.1023/B3AVISI.0000029664.99615.94.

[5] S. Lee, D.-M. Kwak, "A terrain classification method for ugv autonomous navigation based on surf," *URAI 2011 - 2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence*, Nov. 2011. DOI: 10.1109/URAI.2011.6145981.

[6] G. Chowdhary, E. Johnson, D. Magree, A. Wu, A. Shein, "Gps-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft," *Journal of Field Robotics*, vol. 30, May 2013. DOI: 10.1002/rob.21454.

[7] W. Liu, Y. Wu, "Novel technique for vision-based uav navigation," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 47, pp. 2731–2741, Nov. 2011. DOI: 10.1109/TAES.2011.6034661.

[8] J. Zhang, Y. Wu, W. Liu, X. Chen, "Novel approach to position and orientation estimation in vision-based uav navigation," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 46, pp. 687–700, May 2010. DOI: 10.1109/TAES.2010.5461649.

[9] C. Schlaile, O. Meister, N. Frietsch, C. Kessler, J. Wendel, G. Trommer, "Using natural features for vision based navigation of an indoor-vtol mav," *Aerospace Science and Technology*, vol. 13, pp. 349–357, Oct. 2009. DOI: 10.1016/j.ast.2009.09.001.

[10] F. Bonin-Font, A. Ortiz, G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of Intelligent and Robotic Systems*, vol. 53, pp. 263–296, Nov. 2008. DOI: 10.1007/s10846-008-9235-4.

[11] A. Dawadee, J. Chahl, N. Nandagopal, “An algorithm for autonomous aerial navigation using landmarks,” *Journal of Aerospace Engineering*, vol. 29, p. 04 015 072, Oct. 2015. DOI: 10.1061/(ASCE)AS.1943-5525.0000554.

[12] H. GENCO, B. GÜRKAN, U. SAKARYA, V. E. ÖMÜRLÜ, “Image processing-based navigation for unmanned aerial vehicles,” in *ICASEM 4th International Congress of Applied Sciences, Engineering and Mathematics, Tekirdağ, Türkiye, October 20-23, 2022*.

[13] Y. Li, Q. Li, Y. Liu, W. Xie, “A spatial-spectral sift for hyperspectral image matching and classification,” *Pattern Recognition Letters*, vol. 127, pp. 18–26, 2019, Advances in Visual Correspondence: Models, Algorithms and Applications (AVC-MAA), ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2018.08.032. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865518305117.

[14] W. Kwankhoom, J. Ieamsaard, N. Anupong, “Comparative study of sift and surf algorithm for traditional thai painting recognition,” *ICIC International*, vol. 15, pp. 617–625, 2021, ISSN: 1881-803X. DOI: 10.24507/icicel.15.06.617. [Online]. Available: http://www.icicel.org/ell/contents/2021/6/el-15-06-10.pdf.

[15] D. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[16] J. Luo, G. Oubong, “A comparison of sift, pca-sift and surf,” *International Journal of Image Processing*, vol. 3, Oct. 2009.

[17] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, “Speeded-up robust features (surf),” *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, Jun. 2008. DOI: 10.1016/j.cviu.2007.09.014.

[18] Y. Kim, H. Bang, *Introduction to Kalman Filter and Its Applications*, F. Govaers, Ed. Rijeka: IntechOpen, 2018, ch. 2. DOI: 10.5772/intechopen.80600. [Online]. Available: https://doi.org/10.5772/intechopen.80600.

[19] M. H. Haider, T. Islam, M. Islam, M. Shajid-Ul-Mahmud, “Comparison of complementary and kalman filter based data fusion for attitude heading reference system,” vol. 1919, Feb. 2017. DOI: 10.1063/1.5018520.

[20] R. Labbe, *Kalman and bayesian filters in python*, https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python, 2014.

[21] MATLAB, *9.12.0.1956245 (R2022a)*. Natick, Massachusetts: The MathWorks Inc., 2022.

[22] *Estimate geometric transform from matching point pairs (msac)*. [Online]. Available: https://www.mathworks.com/help/vision/ref/estimategeometrictransform.html.

[23] AGISOFT, *1.8.0 build 13597*. [Online]. Available: http://www.agisoft.com/downloads/installer/.

[24] Agisoft, *Orthomosaic & dem generation (without gcps)*. [Online]. Available: https://agisoft.freshdesk.com/support/solutions/articles/31000157908-orthomosaic-dem-generation-without-gcps-.

[25] *Finding matching features*. [Online]. Available: https://www.mathworks.com/help/vision/ref/matchfeatures.html.

[26] *Find image rotation and scale using automated feature matching*. [Online]. Available: https://www.mathworks.com/help/vision/ug/find-image-rotation-and-scale-using-automated-feature-matching.html.

[27] Unity, *(lts 2019.4.32f1) free version*, 2022. [Online]. Available: https://unity.com/download.

[28] Google Earth, MAXAR Technologies, *YTÜ Stadyumu ( 41°01'32"N 28°53'09"E)*, Accessed: 2021-04-10. [Online]. Available: https://earth.google.com/web/@41.02565315,28.88591079,62.35477783a,147.28075719d,35y,103.62879863h,0.27796297t,0r.

# PUBLICATIONS FROM THE THESIS

## Conference Papers

1. H. Genco, B. Gūrkan, U. Sakarya, V.E. Ömürlü, “Image Processing-Based Navigation for Unmanned Aerial Vehicles”, ICASEM 4th International Congress of Applied Sciences, Engineering and Mathematics, Tekirdağ, Türkiye, October 20-23, 2022