electronics

MDPI

Article

EKF-SLAM for Quadcopter Using Differential Flatness-Based LQR Control

Shyam Rauniyar [1], Sameer Bhalla [2], Daegyun Choi [1] and Donghoon Kim [1,*]

[1] Department of Aerospace Engineering & Engineering Mechanics, University of Cincinnati, Cincinnati, OH 45221, USA
[2] Department of Mechanical & Materials Engineering, University of Cincinnati, Cincinnati, OH 45221, USA
* Correspondence: donghoon.kim@uc.edu

Abstract: SLAM (Simultaneous Localization And Mapping) in unmanned aerial vehicles can be an advantageous proposition during dangerous missions where aggressive maneuvers are paramount. This paper proposes to achieve it for a quadcopter using a differential flatness-based linear quadratic regulator while utilizing sensor measurements of an inertial measurement unit and light detection and ranging considering sensors' constraints, such as a limited sensing range and field of view. Additionally, a strategy to reduce the computational effort of Extended Kalman Filter-based SLAM (EKF-SLAM) is proposed. To validate the performance of the proposed approach, this work considers a quadcopter traversing an 8-shape trajectory for two scenarios of known and unknown landmarks. The estimation errors for the quadcopter states are comparable for both cases. The accuracy of the proposed method is evident from the Root-Mean-Square Errors (RMSE) of 0.04 m, 0.04 m/s, and 0.34 deg for the position, velocity, and attitude estimation of the quadcopter, respectively, including the RMSE of 0.03 m for the landmark position estimation. Lastly, the averaged computational time for each step of EKF-SLAM with respect to the number of landmarks can help to strategically choose the respective number of landmarks for each step to maximize the use of sensor data and improve performance.

Keywords: quadcopter; simultaneous localization and mapping; extended Kalman filter; linear quadratic regulator; differential flatness

Citation: Rauniyar, S.; Bhalla, S.; Choi, D.; Kim, D. EKF-SLAM for Quadcopter Using Differential Flatness-Based LQR Control. *Electronics* **2023**, *12*, 1113. https://doi.org/10.3390/electronics12051113

Academic Editor: Quang Ha

Received: 21 January 2023
Revised: 9 February 2023
Accepted: 17 February 2023
Published: 24 February 2023

1. Introduction

SLAM (Simultaneous Localization And Mapping) aims to solve the problem of localization of an autonomous robot in an unknown environment by simultaneously mapping the surroundings. The emergence of indoor applications of mobile robotics has made SLAM an attractive alternative to user-built maps [1]. Moreover, SLAM is gradually finding applications outdoors, underwater, and in space devoid of a Global Positioning System (GPS) [2]. Essentially, SLAM can be divided into a process of front-end and back-end formulation, where the front-end comprises feature extraction, data association, and outlier rejection, whereas the back-end deals with the main estimation process of robot poses and landmark positions [1]. Based on factors, such as spatial representation, the structure and dynamics of the environment, and the employed sensors, a variety of approaches to SLAM can be used, which include but are not limited to GraphSLAM, FastSLAM, and VoSLAM [3].

Several developments have been made for mobile mapping systems using ground-based systems, but unmanned aerial vehicles (UAVs) could provide advantages at hazardous sites compared to other platforms and enable mapping at different resolutions with the ability to fly at different elevations and speeds [4]. For UAVs, the most popular method for SLAM implementation has been based on Extended Kalman Filter (EKF), that is EKF-SLAM, with a majority using Inertial Measurement Units (IMUs) as proprioceptive sensors and visual odometry-based sensors such as cameras as exteroceptive sensors [5].

The EKF-SLAM approach was first proposed by Smith et al. [6], where EKF is a Bayesian approach of estimation utilizing posterior probability distribution. EKF-SLAM faces key issues, especially in terms of computational effort as computation increases quadratically with the number of landmarks [7]. Still, EKF-SLAM is the most popular method with the ability to deal with non-linear systems, and plenty of available research and resources [8].

SLAM is basically solved specifically for vision-based slow robots in 2D indoor environments [1], but it is still an enigma for highly agile robots such as quadcopters, operating in indoor or outdoor environments. The SLAM algorithm has been studied for UAVs with 3D observations of LiDAR (Light Detection And Ranging) or comparable to LiDAR in a few research works [2,4,9–12]. Note that cameras are extremely sensitive to lighting conditions and have limitations in capturing the geometry of the observed scene and generating more accurate maps, thus a more effective sensor, LiDAR, is being considered more lately [4]. Cho and Hwang [9] conducted SLAM with just four landmarks using EKF-SLAM and observation in terms of range and attitude difference. In [10], 3D SLAM is performed for UAVs using a linear Kalman filter with the linearized model for the UAVs and range-bearing measurements. A data fusion algorithm was proposed for UAVs by the integration of LiDAR measurements with GPS to perform in GPS-degraded environments [11]. Kim et al. [12] discussed an application of LiDAR for the localization of speedy UAVs in a GPS-denied outdoor environment while conducting surface reconstruction and elevation registration. While all of the aforementioned studies have used EKF in some form, there are others. Sadeghzadeh-Nokhodberiz et al. [2] studied the application of 3D SLAM on UAVs using FastSLAM and LiDAR. Karam et al. [4] used LiDAR for validation of the performance of GraphSLAM using rangefinders on UAVs.

Most works discussed above are focused on localization and mapping and mostly have not discussed the control strategies used. Many of them used the most basic controller such as PID control with pre-defined control signals for following a set trajectory. In [13], SLAM was achieved using a sliding mode control for the non-linear system of a quadcopter, but still lacked the ability to closely follow a given trajectory over extended periods. Moreover, UAVs can have aggressive maneuvers where fast and reliable SLAM is required. Thus, this work proposes the use of a Differential Flatness-based Linear Quadratic Regulator (DF-LQR). It is known that LQR is an optimum control superior to other classical approaches including PID in terms of robustness, energy performance, computational efficiency, and tuning parameters [14]. However, LQR by itself can only be applied to linear or linearized systems. A DF-based approach, where the non-linear system of quadcopters can be formulated linearly for control using LQR, has been proposed to achieve aggressive maneuvers [15]. For UAVs operating in any dangerous conditions, aggressive maneuvers could be paramount and thus DF-based LQR control for quadcopters could be an ideal choice. In practice, control is achieved in SLAM based on control signals obtained using estimated states rather than true-value states, but many researchers have been conducting studies assuming true-value states [2,9,10]. DF-based LQR control has not been studied in conjunction with 3D SLAM, and thus this work proposes to do so while also calculating control signals from estimated states. The effectiveness of this control strategy is validated by comparing the results of estimated quadcopter states when landmarks are unknown and are also being estimated to a case when landmarks are known.

As mentioned previously, EKF-SLAM is computationally intensive with the increasing number of landmarks, but it is still a method of choice for several researchers, as is evident from several kinds of research over the years using EKF [8]. While there has been research on reducing the computational complexity of EKF-SLAM with the use of local maps for 2D motion [16], it would be beneficial if there are metrics that can provide inference on computation time for different steps of SLAM, such as prediction, update, and registration such that decisions can be made on the size of map space for predictions, the number of landmarks for update, or registration, or even the time step for SLAM. This work proposes a strategy to reduce the computational effort of EKF-SLAM, which is tightly combined with the UAV control discussed above. It conducts multiple simulations over different numbers

of landmarks to study the relationship between these time parameters with the landmarks in consideration. Although these metrics were derived for the specific computing resources available in this study, comparable metrics can be derived for other computing resources in use. Therefore, it is thought that it can provide direction to researchers who want to use or improve EKF-SLAM in the future.

In summary, the contributions of this paper are summarized below:

1. Study of DF-based LQR control in 3D EKF-SLAM to achieve a better and more realistic trajectory control using estimated states.
2. A pragmatic way of using LiDAR measurements as they become available based on UAV position and LiDAR's Field Of View (FOV).
3. Inducing decision-making to improve the computational efficiency of the appropriate number of landmarks or EKF-SLAM through the development of a metric of the relationship between the number of landmarks and the time step.

2. Methodologies

2.1. Dynamic and Kinematic Model of Quadcopters

The inertial and body frames and the state variables to describe the motion of the quadcopter are depicted in Figure 1. The inertial frame N, which is fixed to the surface of the Earth at mean sea level, is defined as $\hat{\mathbf{n}}_i$ for $i = 1, 2$, and 3, where $\hat{\mathbf{n}}_1$ is directed towards North, $\hat{\mathbf{n}}_2$ is directed towards East, and $\hat{\mathbf{n}}_3$ points to the center of the Earth. Note that one assumes a flat, non-rotating Earth with a uniform gravitational field. The body frame B, whose origin is fixed to the center of gravity of the quadcopter, is defined as $\hat{\mathbf{b}}_i$ for $i = 1, 2$, and 3, where $\hat{\mathbf{b}}_1$ is directed normal to the field of view of the LiDAR on the quadcopter, $\hat{\mathbf{b}}_2$ points to the direction of the right arm of the quadcopter, and $\hat{\mathbf{b}}_3$ is directed downwards, which is perpendicular to the quadcopter plane.

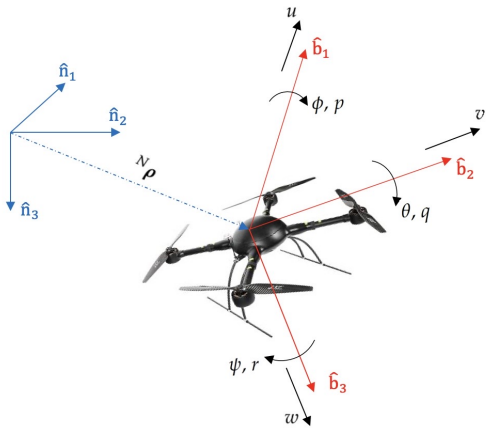

Figure 1. Definitions of the reference frames and the state variables of the quadcopter.

The quadcopter is assumed to be an exactly symmetrical rigid body, which implies that the inertia matrix is diagonal, with 3 Degrees-Of-Freedom (DOF) translation motion and 3 DOF rotational motion. The quadcopter's attitude with respect to the inertial frame is expressed using the 3-2-1 set of the Euler angles as ϕ, θ, and ψ that represent the roll, pitch, and yaw angles, respectively. Note that the direction cosine matrix that maps from the body frame to the inertial frame is described as [17]:

$$C_{NB} = C_3(\psi)\, C_2(\theta)\, C_1(\phi). \tag{1}$$

Note that each direction cosine matrix is defined as

$$C_3(\psi) = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, C_2(\theta) = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}, C_1(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}.$$

where, for instance, $c\psi$ and $s\psi$ are $\cos\psi$ and $\sin\psi$, respectively.

To describe the motion of the quadcopter, this work defines a state model as [18]:

$$\begin{bmatrix} {}^{N}\dot{\boldsymbol{\rho}} \\ {}^{B}\dot{\boldsymbol{\nu}} \\ {}^{B}\dot{\boldsymbol{\Lambda}} \\ {}^{B}\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} C_{NB}\,{}^{B}\boldsymbol{\nu} \\ -{}^{B}\boldsymbol{\omega} \times {}^{B}\boldsymbol{\nu} + \left(C_{NB}^{\mathrm{T}}\,{}^{N}\mathbf{f}_{g} + {}^{B}\mathbf{f}_{T}\right)/m \\ D\,{}^{B}\boldsymbol{\omega} \\ J^{-1}\left[-{}^{B}\boldsymbol{\omega} \times \left(J\,{}^{B}\boldsymbol{\omega}\right) + {}^{B}\boldsymbol{\tau}\right] \end{bmatrix}, \tag{2}$$

where ${}^{N}\boldsymbol{\rho} \in \mathbb{R}^3$ is the position of the quadcopter in the inertial frame in Cartesian coordinates, ${}^{B}\boldsymbol{\nu} \in \mathbb{R}^3$ is the velocity of the quadcopter expressed in the body frame, ${}^{B}\boldsymbol{\Lambda} = [\phi,\ \theta,\ \psi]^{\mathrm{T}} \in \mathbb{R}^3$ is the vector composed of the 3-2-1 set of the Euler angles, ${}^{B}\boldsymbol{\omega} = [p,\ q,\ r]^{\mathrm{T}} \in \mathbb{R}^3$ is the angular velocity of the quadcopter expressed in the body frame, ${}^{N}\mathbf{f}_{g}$ is the gravity force defined by $g\hat{\mathbf{n}}_3$, g is the gravitational constant, ${}^{B}\mathbf{f}_{T}$ is the thrust force defined by $-f_T\hat{\mathbf{b}}_3$, f_T is the intensity of the thrust, m is the mass of the quadcopter, $J \in \mathbb{R}^{3\times3}$ is the moment of inertia of the quadcopter, ${}^{B}\boldsymbol{\tau} = \left[\tau_\phi,\ \tau_\theta,\ \tau_\psi\right]^{\mathrm{T}} \in \mathbb{R}^3$ is the torque acting on the quadcopter in the body frame, and the matrix D is defined as:

$$D = \frac{1}{\cos\theta}\begin{bmatrix} \cos\theta & \sin\phi\sin\theta & \cos\phi\sin\theta \\ 0 & \cos\phi\cos\theta & -\sin\phi\cos\theta \\ 0 & \sin\phi & \cos\phi \end{bmatrix}. \tag{3}$$

Note that the top-left superscripts represent the frame in which the components of each vector are defined. From here on, for simplicity, the superscripts N and B are omitted unless otherwise specified for clarity. Additionally, the aerodynamic and gyroscopic effects of the rotors are ignored to reduce the complexity.

2.2. Trajectory Control

A quadcopter is an under-actuated system that uses 4 independent rotors to control 6 DOF motion. For this reason, the position control of the quadcopter is achieved by changing its attitude and thrust. Usually, a trajectory controller is used as an outer loop control while the attitude controller plays a role in an inner loop control [19]. However, the quadcopter may suffer from large tracking errors when following high-speed trajectories. This problem is countered by using the concept of DF.

Figure 2 shows the procedure of the trajectory control for the quadcopter in this work. Once the desired states ($\mathbf{r}_d$) and desired input ($\mathbf{u}_d$) using DF are determined for a given trajectory ($\mathbf{y}_d$), the LQR-based trajectory controller generates an output ($\mathbf{u}$) that is used in an inverse mapping process, at the end of which we obtain the required thrust information (f_T), desired attitude ($\boldsymbol{\Lambda}_d = [\phi_d,\ \theta_d,\ \psi_d]^{\mathrm{T}}$), and desired angular velocity ($\boldsymbol{\omega}_d = [p_d,\ q_d,\ r_d]^{\mathrm{T}}$). The desired attitudes and angular velocity are utilized to compute the control torque ($\boldsymbol{\tau}$) via the attitude controller. Then, the obtained f_T and $\boldsymbol{\tau}$ are used to control the quadcopter. Each of the steps of the trajectory control is explained in the following subsections.

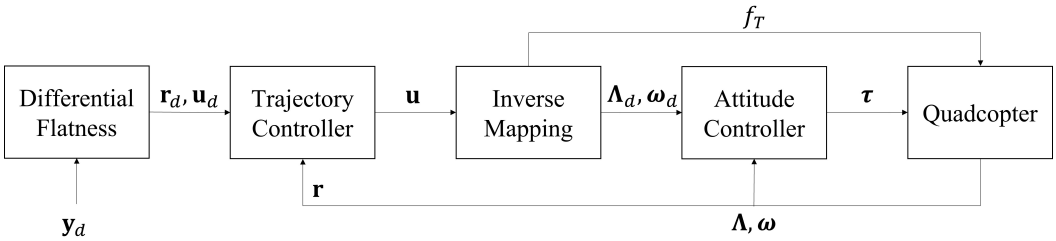

Figure 2. Flowchart of the trajectory control.

2.2.1. Differential Flatness

DF is a structural property of a class of nonlinear dynamical systems, denoting that all system variables (such as state vectors and control inputs) can be written in terms of a set of specific variables (called flat outputs) and their derivatives [15,20]. In other words, a flat output is expressed as $\mathbf{y}_d = \mathbf{f}_y(\mathbf{r}_d,\ \mathbf{u}_d)$, where $\mathbf{r}_d$ and $\mathbf{u}_d$ can be expressed as $\mathbf{r}_d = \mathbf{g}_r(\mathbf{y}_d, \dot{\mathbf{y}}_d, \ddot{\mathbf{y}}_d, \ldots)$ and $\mathbf{u}_d = \mathbf{g}_u(\mathbf{y}_d, \dot{\mathbf{y}}_d, \ddot{\mathbf{y}}_d, \ldots)$ if they exist [15].

With this concept, this work defines $\mathbf{y}_d = [\boldsymbol{\rho}_d^{\mathrm{T}},\ \psi_d]^{\mathrm{T}} \in \mathbb{R}^4$, where $\boldsymbol{\rho}_d \in \mathbb{R}^3$ and ψ_d represent the position of the desired trajectory in the inertial frame and the heading angle of the desired trajectory, respectively. Thus, the corresponding $\mathbf{r}_d$ can be defined in terms of flat outputs as

$$\mathbf{r}_d = \left[\boldsymbol{\rho}_d^{\mathrm{T}},\ \dot{\boldsymbol{\rho}}_d^{\mathrm{T}},\ \psi_d\right]^{\mathrm{T}} \in \mathbb{R}^7. \tag{4}$$

Likewise, $\mathbf{u}_d$ is defined in terms of flat outputs $\ddot{\boldsymbol{\rho}}_d$ and $\dot{\psi}_d$ as

$$\mathbf{u}_d = \left[[\ddot{\boldsymbol{\rho}}_d - g\hat{\mathbf{n}}_3\,]^{\mathrm{T}},\ \dot{\psi}_d\right]^{\mathrm{T}} \in \mathbb{R}^4. \tag{5}$$

The desired states can also be found from the following:

$$\dot{\mathbf{r}}_d = A\mathbf{r}_d + B\mathbf{u}_d + \mathbf{b}g, \tag{6}$$

where

$$A = \begin{bmatrix} 0_{3\times3} & I_{3\times3} & \mathbf{0}_{3\times1} \\ & 0_{4\times7} & \end{bmatrix} \in \mathbb{R}^{7\times7},\ B = \begin{bmatrix} 0_{3\times4} \\ I_{4\times4} \end{bmatrix} \in \mathbb{R}^{7\times4},\ \mathbf{b} = \begin{bmatrix} \mathbf{0}_{5\times1} \\ 1 \\ 0 \end{bmatrix} \in \mathbb{R}^7.$$

Here, $0_{i\times j}$ represents a zero matrix with a dimension of $\mathbb{R}^{i\times j}$, $\mathbf{0}_{i\times1}$ represents a zero vector with a dimension of $\mathbb{R}^{i\times1}$, and $I_{k\times k}$ represents an identity matrix with the dimension of $\mathbb{R}^{k\times k}$.

2.2.2. LQR-Based Trajectory Controller

This work considers an LQR-based control that can provide trajectory control by linearizing a nonlinear system at a nominal trajectory [21]. Since the position, velocity, and heading angle of the quadcopter are the main states out of DF, the trajectory control considers the following state vector:

$$\mathbf{r} = \left[\boldsymbol{\rho}^{\mathrm{T}},\ \dot{\boldsymbol{\rho}}^{\mathrm{T}},\ \psi\right]^{\mathrm{T}} \in \mathbb{R}^7. \tag{7}$$

The linearized state space model is obtained as [15]:

$$\dot{\mathbf{r}} = A\mathbf{r} + B\mathbf{u} + \mathbf{b}g, \tag{8}$$

where $\mathbf{u}$ is defined by

$$\mathbf{u} \equiv \begin{bmatrix} \mathbf{u}_p \\ u_\psi \end{bmatrix} = \begin{bmatrix} \dfrac{1}{m} C_{NB}\,\mathbf{f}_T \\ \dot{\psi} \end{bmatrix} \in \mathbb{R}^4. \tag{9}$$

Note that $\mathbf{u}_p$ is expressed in the inertial frame.

To find the optimal control input that follows the desired trajectory, one considers the following performance index [19]:

$$\mathcal{L} = \int_0^\infty \left(\tilde{\mathbf{r}}^{\mathrm{T}}\tilde{Q}\tilde{\mathbf{r}} + \tilde{\mathbf{u}}^{\mathrm{T}}\tilde{R}\tilde{\mathbf{u}}\right)\mathrm{d}t, \tag{10}$$

subject to

$$\dot{\tilde{\mathbf{r}}} = A\tilde{\mathbf{r}} + B\tilde{\mathbf{u}}, \tag{11}$$

where $\tilde{\mathbf{r}} \in \mathbb{R}^7$ is the state error defined as $\tilde{\mathbf{r}} = \mathbf{r} - \mathbf{r}_d$, $\tilde{\mathbf{u}} \in \mathbb{R}^7$ is the difference between the trajectory control input and the desired input defined as $\tilde{\mathbf{u}} = \mathbf{u} - \mathbf{u}_d$, and $\tilde{Q} \in \mathbb{R}^{7\times7}$ and $\tilde{R} \in \mathbb{R}^{4\times4}$ are the positive-definite weight matrices, which are determined by the Bryson's rule [22]. Note that Equation (11) can be obtained from the difference between Equations (6) and (8) because these equations are comparable with the same A, B, and $\mathbf{b}$ coefficients.

The optimal control input is given by [23]

$$\tilde{\mathbf{u}} = -\tilde{R}^{-1}B^{\mathrm{T}}\tilde{P}\tilde{\mathbf{r}}, \tag{12}$$

where $\tilde{P} \in \mathbb{R}^{7\times7}$ is the positive-definite Riccati matrix, which is determined by solving the following algebraic Ricatti equation:

$$A^{\mathrm{T}}\tilde{P} + \tilde{P}A - \tilde{P}B\tilde{R}^{-1}B^{\mathrm{T}}\tilde{P} + \tilde{Q} = 0. \tag{13}$$

Then, $\mathbf{u}$ in Equation (9) is ultimately calculated as:

$$\mathbf{u} \equiv \begin{bmatrix} \mathbf{u}_p \\ u_\psi \end{bmatrix} = \mathbf{u}_d + \tilde{\mathbf{u}}. \tag{14}$$

It is important to note that $\mathbf{u}$ cannot be directly used to track the desired trajectory because a quadcopter is an under-actuated system. Therefore, based on $\mathbf{u}$, the useful expressions using Equations (9) and (14) are described for the attitude control in the following subsection.

2.2.3. Inverse Mapping

The resulting $\mathbf{u}$ in Equation (14) is converted into f_T, the desired attitudes ϕ_d, θ_d, and the desired yaw angular rate r_d through the inverse mapping process. For a stable flight, one would not want continuously changing roll and pitch in a quadcopter. Thus, the desired roll and pitch angular rates in the body frame are considered to be zero, that is, $p_d = 0$ and $q_d = 0$. Additionally, ψ_d is obtained from $\mathbf{y}_d$.

First, $\mathbf{u}_p$ component of Equation (9) can be rearranged as:

$$f_T = m\sqrt{\mathbf{u}_p^{\mathrm{T}}\mathbf{u}_p}. \tag{15}$$

Here, f_T is calculated with $\mathbf{u}_p$ obtained from Equation (14). Additionally, the $\mathbf{u}_p$ component of Equation (9) can be rearranged with desired attitudes as follows:

$$\mathbf{z} \equiv -\frac{m}{f_T}[C_3(\psi_d)]^{\mathrm{T}}\mathbf{u}_p = C_2(\theta_d)\, C_1(\phi_d)\, \hat{\mathbf{b}}_3, \tag{16}$$

where $\mathbf{z} = [z_1,\ z_2,\ z_3]^{\mathrm{T}} \in \mathbb{R}^3$. Substituting the values of $\mathbf{u}_p$ determined from Equation (14) into Equation (16), $\mathbf{z}$ is calculated using the first part of the equation. Then, the second part of the equation is solved for ϕ_d and θ_d as follows:

$$\phi_d = \sin^{-1}(-z_2),\ \theta_d = \tan^{-1}\left(\frac{z_1}{z_3}\right). \tag{17}$$

Following the expansion of Equation (2) for $\dot{\psi}$ in $\dot{\mathbf{\Lambda}}$ equation, one can have the following relation:

$$\dot{\psi} = r\frac{\cos\phi + q\sin\phi}{\cos\theta}. \tag{18}$$

Since, $\dot{\psi} = u_\psi$ in Equation (9), substituting $\dot{\psi}$ with u_ψ obtained from Equation (14), ϕ, θ with ϕ_d, θ_d from Equation (17), and q with the desired pitch angular rate of $q_d = 0$, the desired yaw angular rate is determined as:

$$r_d = u_\psi \frac{\cos\theta_d}{\cos\phi_d}. \tag{19}$$

2.2.4. Attitude Controller

For attitude control, a PD controller is considered in this work. The attitude controller determines the torque $\boldsymbol{\tau}$ based on the desired attitude and the desired angular rate. The torque components in the body frame at a given instant are calculated as:

$$\begin{aligned} \tau_\phi &= -k_{p_\phi}(\phi - \phi_d) - k_{d_\phi}(p - p_d), \\ \tau_\theta &= -k_{p_\theta}(\theta - \theta_d) - k_{d_\theta}(q - q_d), \\ \tau_\psi &= -k_{p_\psi}(\psi - \psi_d) - k_{d_\psi}(r - r_d). \end{aligned} \tag{20}$$

where k_{p_ϕ}, k_{p_θ}, and k_{p_ψ} are the proportional gains and k_{d_ϕ}, k_{d_θ}, and k_{d_ψ} are the derivative gains.

2.3. EKF-SLAM

The EKF-SLAM estimates the quadcopter states while mapping the estimated landmarks' position simultaneously. This work assumes that the quadcopter has a 3D LiDAR that can provide range-bearing measurements for each identified landmark and an IMU that provides a 3-axis gyroscope and 3-axis accelerometer. For the LiDAR measurement, it is assumed that the observed landmarks are perfectly identified whenever observed using LiDAR providing range-bearing measurements of each identified landmark. In the case of the IMU measurement, especially the accelerometer, the only vertical accelerometer data in the body frame will be used because both planar axes data should ideally be zero in the quadcopter dynamics [24]. Additionally, one assumes that the IMU provides the estimated measurements that are already filtered.

One part of the total state vector estimated by the EKF-SLAM is the quadcopter's state defined by $\mathbf{s} = \left[\boldsymbol{\rho}^\mathrm{T},\ \boldsymbol{\nu}^\mathrm{T},\ \boldsymbol{\Lambda}^\mathrm{T}\right]^\mathrm{T} \in \mathbb{R}^9$, which includes the position in the inertial frame, velocity in the body frame, and attitude in terms of Euler angles.

Unlike the conventional EKF, the total state vector in the EKF-SLAM contains more states, which are the position (i.e., the inertial coordinate) of mapped landmarks defined by $\mathbf{m} = \left[\mathbf{l}_1^\mathrm{T},\ \ldots,\ \mathbf{l}_{n_m}^\mathrm{T}\right]^\mathrm{T} \in \mathbb{R}^{3n_m}$. Here, the subscripts represent landmark IDs associated with mapped landmarks in a set $M = \{1,\ \ldots,\ n_m\}$, and n_m is the total number of mapped landmarks at any given instant. Therefore, the total state vector is defined as:

$$\mathbf{x} = \left[\mathbf{s}^\mathrm{T},\ \mathbf{m}^\mathrm{T}\right]^\mathrm{T} \in \mathbb{R}^{9+3n_m}. \tag{21}$$

Note that the size of the total state vector keeps changing at each time step because the number of mapped landmarks varies.

In addition to the mapped landmarks, there are observed landmarks defined by a set O as shown in Figure 3. If the LiDAR observes n_o landmarks, some may already be mapped (n_κ), but some might not yet have been mapped (n_υ). The observed mapped landmarks are identified as $\kappa \subseteq M$, and the observed but unmapped landmarks are identified with new IDs as $\upsilon = \{n_m + 1,\ n_m + 2,\ \ldots,\ n_m + n_\upsilon\}$, where $\upsilon \nsubseteq M$.

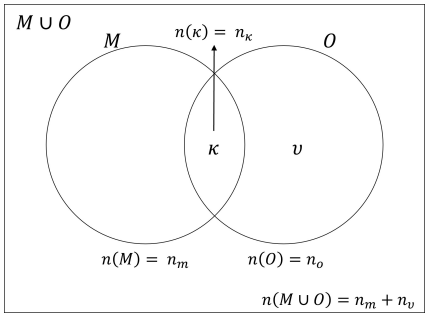

Figure 3. Classification of landmarks.

Moreover, the difference between the conventional EKF and the EKF-SLAM is the registration process of new landmarks. Once unmapped landmarks, which are not incorporated in the total state vector, are observed by LiDAR, they are registered in the total state vector. This process will be explained in Section 2.3.4. An outline of the EKF-SLAM process proposed including the registration process is depicted in Figure 4. The process takes inspiration from [25] proposed for 2D SLAM and is improved to match the continuous non-linear system of the quadcopter in 3D space.

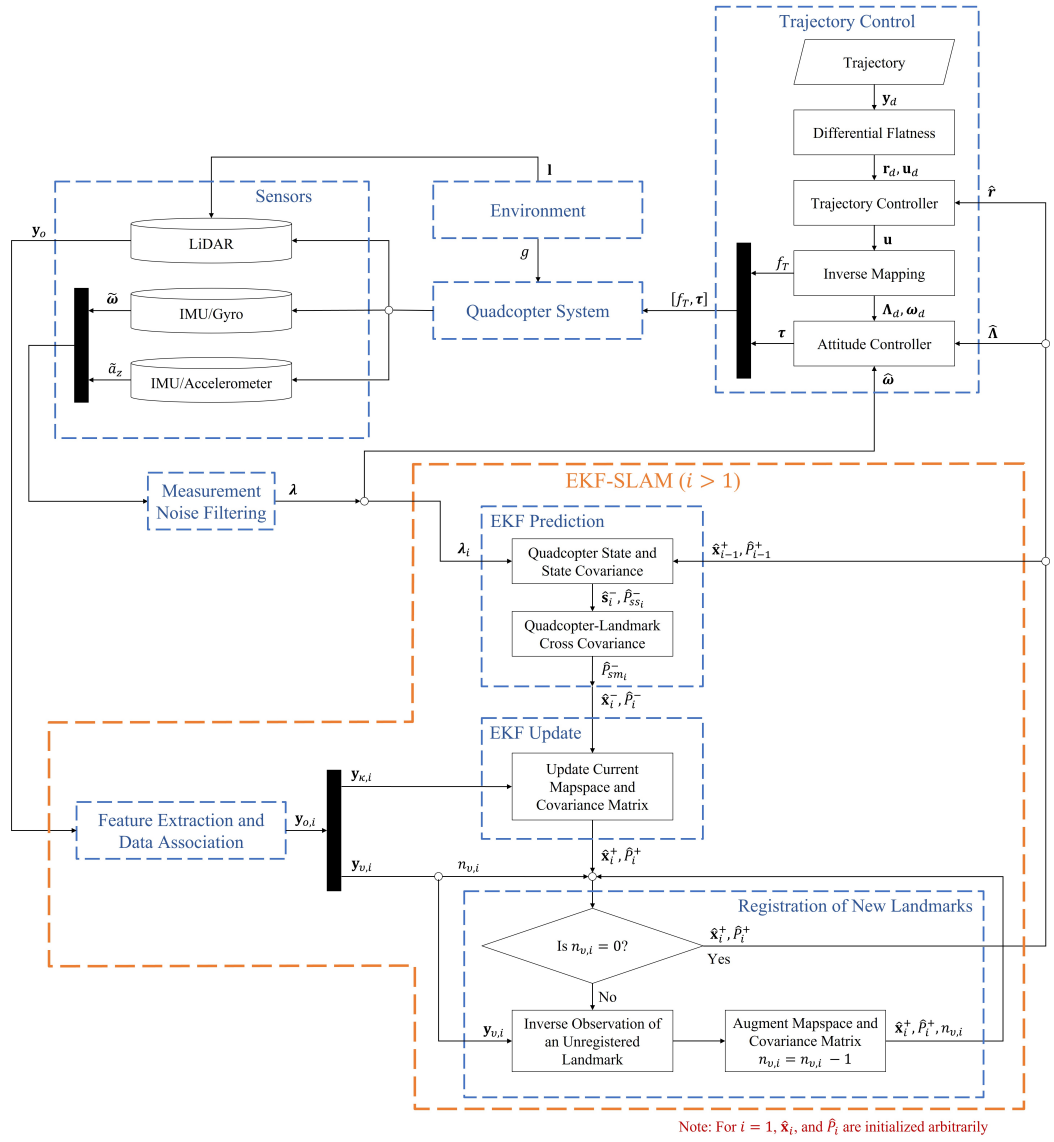

Figure 4. EKF-SLAM flowchart in the sensing, navigation, and control system.

2.3.1. State and Covariance Initialization

The estimated total states $\hat{\mathbf{x}}$ and the state error covariance matrix P are expressed as [7,26]:

$$\hat{\mathbf{x}} = \left[\hat{\mathbf{s}}^{\mathrm{T}},\ \hat{\mathbf{m}}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{9+3n_m}, \tag{22}$$

$$P = \left[\begin{array}{c|c} P_{ss} & P_{sm} \\ \hline P_{ms} & P_{mm} \end{array}\right] = \left[\begin{array}{c|ccc} P_{ss} & P_{sl_1} & \cdots & P_{sl_{n_m}} \\ \hline P_{l_1 s} & P_{l_1 l_1} & \cdots & P_{l_1 l_{n_m}} \\ \vdots & \vdots & \ddots & \vdots \\ P_{l_{n_m} s} & P_{l_{n_m} l_1} & \cdots & P_{l_{n_m} l_{n_m}} \end{array}\right] \in \mathbb{R}^{(9+3n_m)\times(9+3n_m)}, \tag{23}$$

where the hat operator $\hat{()}$ signifies the corresponding estimated states, P_{ss} and P_{mm} are the estimated covariance matrix of the quadcopter and the mapped landmark states, respectively, and P_{sm} and P_{ms} are the estimated cross-covariance matrix between the quadcopter states ($\mathbf{s}$) and the mapped landmarks ($\mathbf{m}$). Note that the subscripts for the covariance matrix P are so named to identify the rows and columns considered. For instance, ss identifies the matrix of the first 9 rows and columns corresponding to the quadcopter's 9 states, and sm identifies the matrix of the first 9 rows and $3n_m$ columns corresponding to the mapped landmark states. In this work, it is assumed that the quadcopter starts without sensing any landmarks. Therefore, $\hat{\mathbf{x}}_1 = \hat{\mathbf{s}}$ and $P_1 = P_{ss}$ are only initialized at the first time step (i.e., $i = 1$).

2.3.2. Prediction of Quadcopter States

This work utilizes Equation (2) to describe the motion of the quadcopter. However, since this mathematical model may not fully describe the actual motion of the quadcopter, the quadcopter model for EKF includes an additional term of the process noise, which is defined by $\mathbf{w} \sim N(\mathbf{0}, Q)$, where Q indicates the process noise covariance matrix. Additionally, considering the estimation is for only nine states of the quadcopter, the propagation is achieved using estimated IMU readings as a control input. This modified model is expressed as

$$\dot{\mathbf{s}} = \mathbf{f}_s(\mathbf{s}, \boldsymbol{\lambda}) + \mathbf{w} = \left[\dot{\boldsymbol{\rho}}^{\mathrm{T}},\ \dot{\boldsymbol{\nu}}^{\mathrm{T}},\ \dot{\boldsymbol{\Lambda}}^{\mathrm{T}}\right]^{\mathrm{T}} + \mathbf{w}, \tag{24}$$

where $\boldsymbol{\lambda} = \left[\hat{a}_z,\ \hat{\boldsymbol{\omega}}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^4$. Here, $\hat{a}_z$ and $\hat{\boldsymbol{\omega}}$ are estimated measurements from the IMU sensor post noise filtering actual vertical accelerometer measurement ($\tilde{a}_z$) and gyroscopic measurement ($\tilde{\boldsymbol{\omega}}$), respectively. The considered IMU measurements for the quadcopter are as follows:

$$\begin{bmatrix} \tilde{a}_z \\ \tilde{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} -f_T/m \\ \boldsymbol{\omega} \end{bmatrix} + \boldsymbol{\eta}, \tag{25}$$

where $\eta \sim N(\mathbf{0}, R_\eta)$, $R_\eta \in \mathbb{R}^{4\times4}$ is the measurement covariance matrix for IMU measurements considered.

In the prediction step, only the quadcopter states will be predicted based on a mathematical model as the landmarks are stationary [7]. Therefore, the predicted quadcopter states at the i^{th} time step are obtained by integrating the following equation through the time step:

$$\dot{\hat{\mathbf{s}}} = \mathbf{f}_s(\hat{\mathbf{s}}, \boldsymbol{\lambda}). \tag{26}$$

Similarly, the predicted quadcopter state covariance matrix at the i^{th} time step, $P_{ss} \in \mathbb{R}^{9\times9}$, is obtained by integrating the following equation through the time step:

$$\dot{P}_{ss} = FP_{ss} + P_{ss}F^{\mathrm{T}} + Q, \tag{27}$$

where F is the Jacobian matrix for the state space function with respect to the quadcopter state as:

$$F \equiv \left.\frac{\partial \mathbf{f}_s}{\partial \mathbf{s}}\right|_{\hat{\mathbf{s}},\boldsymbol{\lambda}} \in \mathbb{R}^{9\times 9}. \tag{28}$$

Additionally, the prediction equation for the cross-covariance matrix between the quadcopter states and the mapped landmark states is given by

$$\dot{P}_{sm} = FP_{sm}, \tag{29}$$

and $P_{sm} \in \mathbb{R}^{9\times 3n_m}$ is obtained by integrating Equation (29). Moreover, P_{ms} is simply computed by $P_{ms} = P_{sm}^{\mathrm{T}}$. Note that Equation (29) is a corollary equation for the established linear form of cross-covariance update in SLAM [7,26] so that it embeds the continuous nature of the quadcopter system.

2.3.3. Updating Total State

For an arbitrary observed landmark, the LiDAR provides the azimuth, elevation, and range information, which are defined as α, β, and δ, respectively, shown in Figure 5, and the observation is given by $\mathbf{y}_l = [\alpha,\ \beta,\ \delta]^{\mathrm{T}} \in \mathbb{R}^3$.

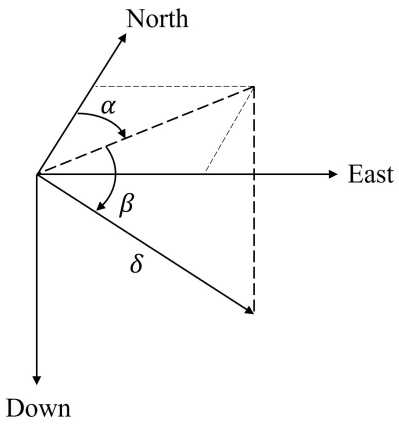

Figure 5. Spherical coordinate representation of LiDAR measurement of a landmark.

If the inertial Cartesian coordinate of the arbitrary observed landmark is $\mathbf{l}$, the observation model for the LiDAR is as follows:

$$\mathbf{y}_l = \mathbf{h}_o(\mathbf{s},\mathbf{l}) + \boldsymbol{\zeta}_l \in \mathbb{R}^3, \tag{30}$$

where $\boldsymbol{\zeta}_l \sim N(\mathbf{0}, R_\zeta), R_\zeta \in \mathbb{R}^{3\times 3}$ is the measurement covariance matrix for LiDAR measurements, and also

$$\mathbf{h}_o(\mathbf{s},\mathbf{l}) = \begin{bmatrix} \tan^{-1}\left(\dfrac{c_3}{\sqrt{c_1^2+c_2^2}}\right) \\ \tan^{-1}\left(\dfrac{c_1}{c_2}\right) \\ \sqrt{c_1^2+c_2^2+c_3^2} \end{bmatrix} \in \mathbb{R}^3, \tag{31}$$

where $\mathbf{c} = \begin{bmatrix} c_1,\ c_2,\ c_3 \end{bmatrix}^{\mathrm{T}} = C_{NB}^{\mathrm{T}}(\mathbf{l}-\boldsymbol{\rho})$ is the Cartesian position of the landmark in the body frame.

The measurements for observed landmarks O are represented as

$$\mathbf{y}_o = \begin{bmatrix} \mathbf{y}_\kappa \\ \mathbf{y}_v \end{bmatrix} \in \mathbb{R}^{3n_o}, \tag{32}$$

where

$$\mathbf{y}_\kappa = \begin{bmatrix} \mathbf{y}_{\kappa,1} \\ \mathbf{y}_{\kappa,2} \\ \vdots \\ \mathbf{y}_{\kappa,n_\kappa} \end{bmatrix} \in \mathbb{R}^{3n_\kappa}, \ \mathbf{y}_v = \begin{bmatrix} \mathbf{y}_{n_m+1} \\ \mathbf{y}_{n_m+2} \\ \vdots \\ \mathbf{y}_{n_m+n_v} \end{bmatrix} \in \mathbb{R}^{3n_v}. \tag{33}$$

Here, for instance, if $\kappa = \{2,4,7\}$, $\mathbf{y}_{\kappa,1}$ signifies the LiDAR observation for the first landmark in κ set, that is the landmark with ID 2. Likewise, $\mathbf{y}_{n_m+1}$ signifies the measurement of an unmapped landmark whose new ID will be $n_m + 1$ post-registration.

It should be noted that only the observed mapped landmarks are used in the observation model considering the requirement of landmark inertial coordinates as one of the arguments in Equation (30). For observed mapped landmarks, one has estimated landmark inertial coordinates. However, for observed but unmapped landmarks that are not the case until the completion of their registration in Section 2.3.4. Hence, the observation function is represented as:

$$\mathbf{y}_\kappa = \mathbf{h}(\mathbf{s}, \mathbf{m}_\kappa) + \mathbf{v} = \begin{bmatrix} \mathbf{h}_o(\mathbf{s}, \mathbf{l}_{\kappa,1}) + \boldsymbol{\zeta}_{\kappa,1} \\ \mathbf{h}_o(\mathbf{s}, \mathbf{l}_{\kappa,2}) + \boldsymbol{\zeta}_{\kappa,2} \\ \vdots \\ \mathbf{h}_o(\mathbf{s}, \mathbf{l}_{\kappa,n_\kappa}) + \boldsymbol{\zeta}_{\kappa,n_\kappa} \end{bmatrix} \in \mathbb{R}^{3n_\kappa}, \tag{34}$$

where

$$\mathbf{v} = \begin{bmatrix} \boldsymbol{\zeta}_{\kappa,1} \\ \vdots \\ \boldsymbol{\zeta}_{\kappa,n_\kappa} \end{bmatrix} \in \mathbb{R}^{3n_\kappa}, \tag{35}$$

is the measurement error vector for the observation function with the measurement covariance matrix $R \in \mathbb{R}^{(3n_\kappa)^2}$.

So, the total measurement vector at the i^{th} time step can be represented as:

$$\mathbf{y}_{o,i} = \begin{bmatrix} \mathbf{y}_{\kappa,i} \\ \mathbf{y}_{v,i} \end{bmatrix} \in \mathbb{R}^{3n_{o,i}}. \tag{36}$$

The first step of updating is to calculate the Kalman gain at the i^{th} time step as:

$$K_i = \hat{P}^-_{(sm,sm_\kappa)_i} H_i^{\mathrm{T}} E_i^{-1} \in \mathbb{R}^{(9+3n_{\kappa,i})\times(3n_{\kappa,i})}, \tag{37}$$

where

$$E_i = H_i \hat{P}^-_{(sm_\kappa,sm_\kappa)_i} H_i^{\mathrm{T}} + R_i \in \mathbb{R}^{(3n_{\kappa,i})^2}, \tag{38}$$

$$\hat{P}^-_{(sm,sm_\kappa)_i} = \begin{bmatrix} \hat{P}^-_{ss} & \hat{P}^-_{sm_\kappa} \\ \hat{P}^-_{ms} & \hat{P}^-_{mm_\kappa} \end{bmatrix}_i \in \mathbb{R}^{(9+3n_{m,i})\times(9+3n_{\kappa,i})}. \tag{39}$$

Here, H_i is a Jacobian matrix of observation function Equation (34), which is calculated as

$$H_i \equiv \left[\frac{\partial \mathbf{h}}{\partial \mathbf{s}}, \frac{\partial \mathbf{h}}{\partial \mathbf{m}_{\kappa,i}} \right] \Bigg|_{\hat{\mathbf{x}}_i^-} \in \mathbb{R}^{(3n_{\kappa,i})\times(9+3n_{\kappa,i})}, \tag{40}$$

and the part of the covariance matrix, $\hat{P}^-_{(ss,sm_\kappa)_i}$, is of the form

$$\hat{P}^-_{(sm_\kappa,sm_\kappa)_i} = \begin{bmatrix} \hat{P}^-_{ss} & \hat{P}^-_{sm_\kappa} \\ \hat{P}^-_{m_\kappa s} & \hat{P}^-_{m_\kappa m_\kappa} \end{bmatrix}_i \in \mathbb{R}^{(9+3n_{\kappa,i})^2}. \tag{41}$$

Then, the correction step based on the observation function Equation (34) at the i^{th} time step can be expressed as:

$$\hat{\mathbf{x}}_i^+ = \hat{\mathbf{x}}_i^- + K_i \left[\mathbf{y}_{\kappa,i} - \mathbf{h}_i \right] \in \mathbb{R}^{9+3n_{\kappa,i}}. \tag{42}$$

Lastly, the updated error covariance matrix at the i^{th} time step is given by [7]

$$\hat{P}_i^+ = \hat{P}_i^- - K_i E_i K_i^{\mathrm{T}} \in \mathbb{R}^{(9+3n_{\kappa,i})^2}. \tag{43}$$

2.3.4. Registration of the New Landmarks

Landmark registration is performed when the quadcopter discovers new unmapped landmarks that are yet to be incorporated into the map space $\mathbf{x}$. Therefore, this operation results in an augmentation of the state vector. All the landmarks observed but unmapped go through this registration step once. During registration, the inertial Cartesian coordinates for each new landmark ($\mathbf{l}$) are determined using the inverse observation model. For an arbitrary landmark observation discussed in Section 2.3.3, the inverse observation model is

$$\mathbf{l} = \mathbf{g}(\mathbf{s}, \mathbf{y}_l) = C_{NB}\, \boldsymbol{\varrho}(\mathbf{y}_l) + \boldsymbol{\rho}, \tag{44}$$

where $\boldsymbol{\varrho}$ is a transformation function that converts LiDAR observations in spherical coordinates to Cartesian coordinates in the body frame as follows:

$$\boldsymbol{\varrho}(\mathbf{y}_l) = \begin{bmatrix} \delta \cos(\beta) \cos(\alpha) \\ \delta \cos(\beta) \sin(\alpha) \\ \delta \sin(\beta) \end{bmatrix}. \tag{45}$$

For any observed unmapped landmark, its registration starts by first calculating its inertial Cartesian coordinate using Equation (44) as:

$$\hat{\mathbf{l}} = \mathbf{g}(\hat{\mathbf{s}}_i^+, \mathbf{y}_l). \tag{46}$$

It is then registered or added into prior updated map space $\hat{\mathbf{x}}^+$ to obtain the augmented map space as:

$$\hat{\mathbf{x}}_i^+ = \begin{bmatrix} \hat{\mathbf{x}}_i^+ \\ \hat{\mathbf{l}} \end{bmatrix} \in \mathbb{R}^{9+3n_m+3}. \tag{47}$$

During registration, the covariance matrix needs augmentation as well with additional rows and columns corresponding to the new landmark. The landmark's covariance is calculated using

$$\hat{P}_{ll} = G_s \hat{P}_{ss_i}^+ G_s^{\mathrm{T}} + G_l R_\zeta G_l^{\mathrm{T}} \in \mathbb{R}^{3\times 3}, \tag{48}$$

where G_s and G_l are Jacobian matrices as follows:

$$G_s = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{s}}\right|_{\hat{\mathbf{s}}_i^+, \mathbf{y}_l} \in \mathbb{R}^{3\times 9},\ G_l = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{y}_l}\right|_{\hat{\mathbf{s}}_i^+, \mathbf{y}_l} \in \mathbb{R}^{3\times 3}. \tag{49}$$

The landmark's cross-covariance with the prior updated map space is determined as [25]:

$$\hat{P}_{lx} = G_s \left[\hat{P}_{ss}^+,\ \hat{P}_{sm}^+\right]_i \in \mathbb{R}^{3\times(9+3n_m)}, \tag{50}$$

$$\hat{P}_{xl} = \hat{P}_{lx}^{\mathrm{T}} \in \mathbb{R}^{(9+3n_m)\times 3}. \tag{51}$$

Finally, the prior updated covariance matrix is augmented as follows:

$$\hat{P}_i^+ = \begin{bmatrix} \hat{P}_i^+ & \hat{P}_{lx} \\ \hat{P}_{xl} & \hat{P}_{ll} \end{bmatrix}_i \in \mathbb{R}^{(9+3n_m+3)^2}. \tag{52}$$

At the end of the landmark registration, the number of mapped landmarks increases by one. That is,

$$n_m = n_m + 1. \tag{53}$$

Figure 6 shows the updated parts of the covariance matrix $\hat{P}$ at different steps of EKF-SLAM.

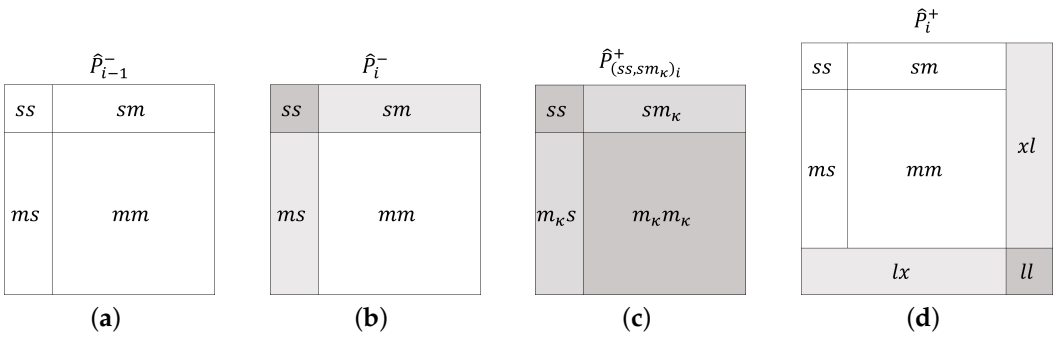

Figure 6. (**a**) Error covariance matrix at the beginning of the i^{th} time step, (**b**) predicted covariance matrix, (**c**) updated part of the covariance matrix, and (**d**) augmented covariance matrix with a single new landmark. Shaded regions are where changes happen. Dark gray represents respective covariance and pale gray represents cross-variance.

If v_i are observed unmapped landmarks at the i^{th} time step corresponding to measurements $\mathbf{y}_{v,i}$ as discussed in Equations (32) and (36), each of the landmarks in v_i is registered sequentially using Equation (46) through (53). After each observed unmapped landmark becomes registered, $n_{v,i}$ reduces by one, that is, $n_{v,i} = n_{v,i} - 1$.

3. Simulation Results and Discussion

For the simulation, the considered quadcopter's mass and inertia properties are tabulated in Table 1. The objective of LQR-based control is to reduce tracking errors while ensuring the least amount of work is performed. The weight matrices are chosen to maintain proximity to the desired trajectory and are tabulated in Table 2. For verifying the efficacy of the DF-LQR-based EKF-SLAM for aggressive maneuvers, a desired trajectory of '8' is chosen. Initially, the quadcopter is set at the origin and controlled to achieve the trajectory over the simulation time of 50 s. Static landmarks are generated randomly around the trajectory of the quadcopter within the region set by the azimuth range, elevation range, and radial range detailed in Table 3. The number of landmarks chosen for simulation is 40 to ensure the observability of the system during the EKF-SLAM process. The sensor specifications assumed for the LiDAR and the IMU equipped in the quadcopter are detailed in Tables 4 and 5, respectively. The specifications listed include the range and noise (standard deviation) for LiDAR measurements and Noise Densities (ND) for accelerometer and gyroscope measurements.

Table 1. Quadcopter properties.

Parameter	Value
Mass, m	1.56 kg
Inertia matrix, J	diag([0.114700, 0.057600, 0.171200])kg-m^2

Table 2. LQR weight matrices.

Parameter	Value
$\tilde{Q}$	diag([0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.04])
$\tilde{R}$	diag([0.2, 0.32, 0.1])

Table 3. Landmark region specifications.

Parameter	Value
Number of landmarks, n	40
Azimuth	$[-180,\ 180]$ deg
Elevation	$[-50,\ 50]$ deg
Range	$[8,\ 20]$ m

Table 4. LiDAR sensor specifications.

Parameter	Value
Azimuth (α) FOV	$[-45,\ 45]$ deg
Elevation (β) FOV	$[-30,\ 30]$ deg
Range (δ)	$[0,\ 100]$ m
Azimuth noise, σ_α	0.33 deg
Elevation noise, σ_β	0.3 deg
Radial noise, σ_δ	0.1 m
Sampling rate	10 Hz

Table 5. IMU sensor specifications.

Parameter	Value
Accelerometer (ND)	300 $\mu g/rtHz^2$
Gyroscope (ND)	0.01 (deg/s)/rtHz
Sampling rate	10 Hz

Figure 7 shows the results of the quadcopter traversing through an 8-shape trajectory using DF-LQR for the two scenarios considered for reviewing the effectiveness of EKF-SLAM. The first scenario is where the landmarks are known, i.e., their inertial positions are known and are used to make the updates during the localization of the quadcopter using EKF. This scenario does not include mapping as the landmarks are already known. The second scenario is where the mapping part of EKF-SLAM comes into effect as the landmarks are unknown. The positions of the landmarks are estimated relative to the initially assumed position of the quadcopter. It can easily be seen from the figure that the case of known landmarks has smoother trajectory control than the unknown case, which is expected.

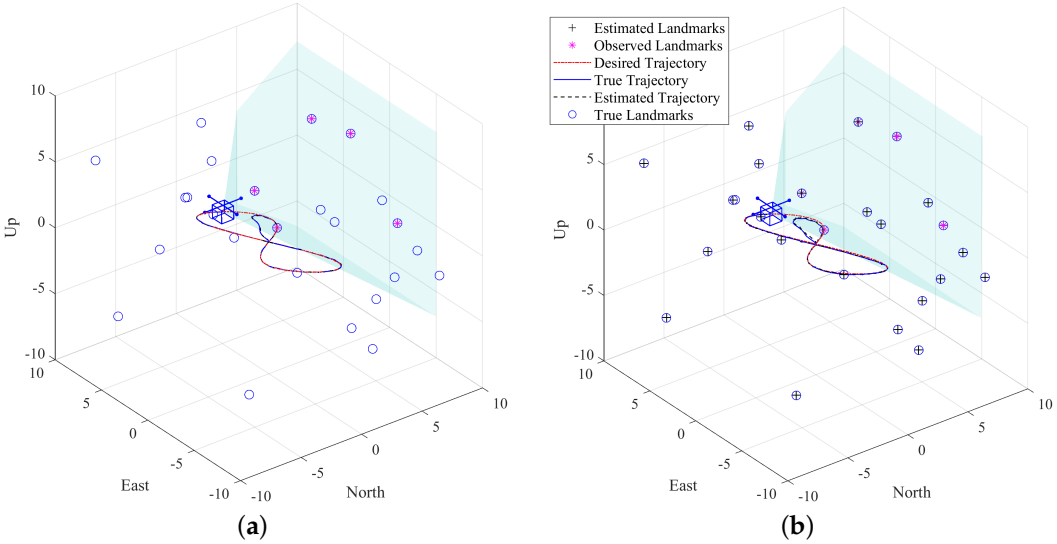

Figure 7. Quadcopter traversing 8-shape trajectory for different scenarios. (**a**) Known landmarks. (**b**) Unknown landmarks.

3.1. Estimated Quadcopter States

Now, one compares the estimated states and errors for each of the scenarios to verify the effectiveness of EKF-SLAM using DF-LQR.

The initial state covariance (P_0) and process noise covariance (Q) matrices considered for each of the scenarios are given in Tables 6 and 7, respectively.

The estimated inertial position of the quadcopter for the known and unknown scenarios are plotted in Figure 8a,b, respectively. It is visible that the estimated positions are closer to the true trajectory in the known landmarks scenario than in the unknown landmarks scenario. It is more evident from the 3σ boundaries for each scenario in Figure 8c,d, where the estimation error is lower for the known landmarks case than the unknown landmarks case. However, this greater error in the unknown landmarks scenario is inevitable considering the process also includes the estimation of landmark positions. Moreover, the average Root-Mean-Square Error (RMSE) over the duration of the simulation for its estimated inertial position hovered around 0.04 m when the landmarks were unknown compared to 0.012 m when the landmarks were known. Thus, the increase in the margin of error for the unknown landmarks case is not substantially larger than for the known landmarks case.

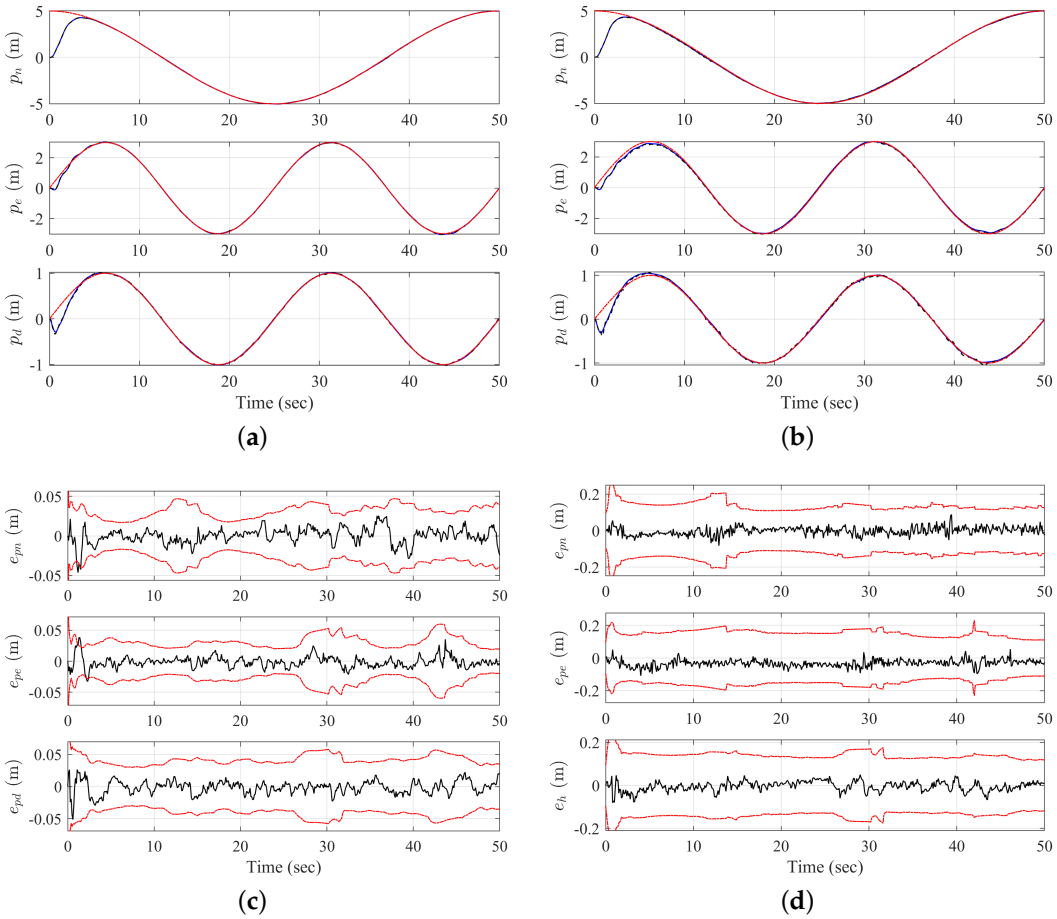

Figure 8. Quadcopter's estimated position and corresponding estimation error when following an 8-shape trajectory for different cases. For (**a**) position for known landmarks and (**b**) position for unknown landmarks, true states (——), estimated states (- - -), and desired states (-·-·-). For (**c**) estimated position error for known landmarks and (**d**) estimated position error for unknown landmarks, estimation error (——), and 3σ boundary (-·-·-).

Table 6. Initial state covariance and process noise covariance matrices for the known landmarks scenario.

Parameter	Value
Initial state covariance (P_0)	diag([0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.02, 0.02, 0.02])
Process noise covariance (Q)	diag([0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.02, 0.02, 0.02])

Table 7. Initial state covariance and process noise covariance matrices for the unknown landmarks scenario.

Parameter	Value
Initial state covariance (P_0)	diag([0.001, 0.001, 0.001, 0.1, 0.1, 0.1, 0, 0, 0])
Process noise covariance (Q)	diag([0.2, 0.2, 0.05, 0.01, 0.01, 0.01, 0.02, 0.2, 0.2])

The same was the case when comparing results for the estimated velocities over the simulation duration of the two scenarios. As expected, when the landmarks were known, the quadcopter motion closely followed the desired velocity trajectory compared to when landmarks were unknown, as evident from Figure 9. The average RMSE for the estimated velocity of the quadcopter when landmarks were known was 0.03 m/s and that for unknown landmarks scenario was 0.04 m/s. Again, the estimations were accurate and not that far apart for the two scenarios. It was actually more erred for the known landmarks scenario, which is due to assumed P_0 and Q matrices.

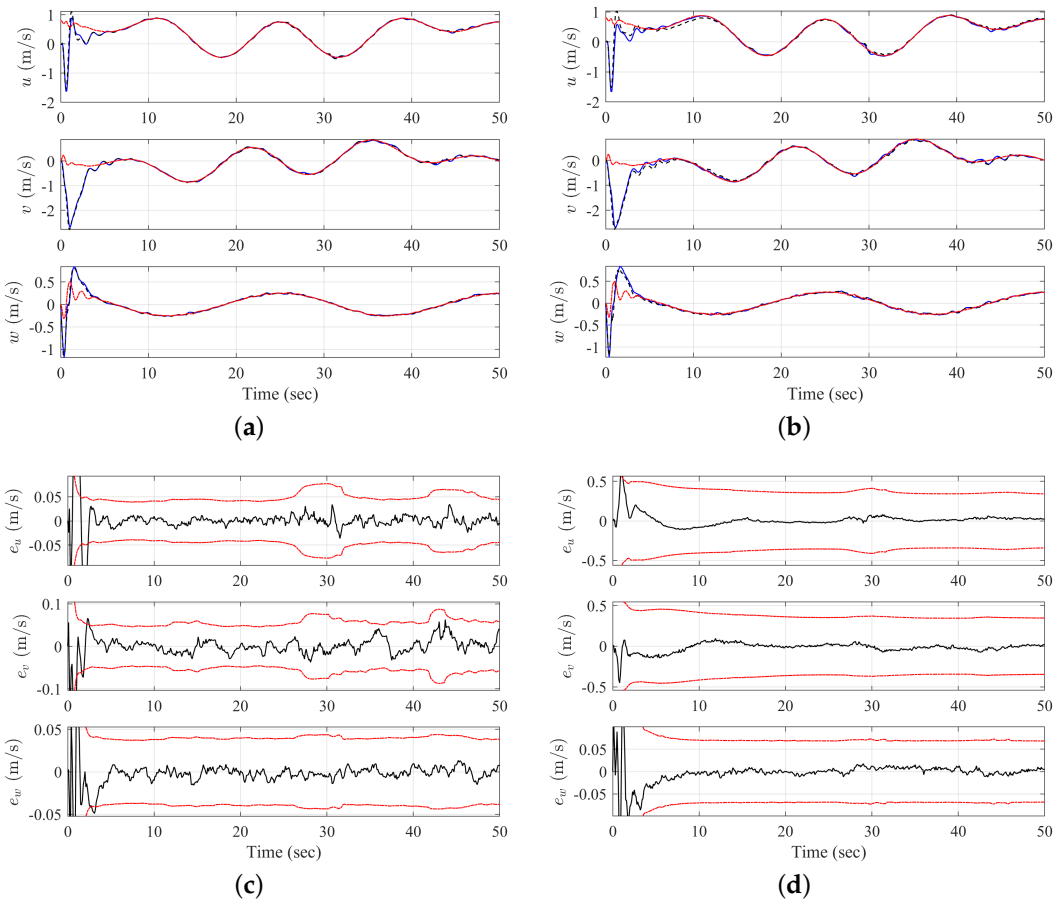

Figure 9. Quadcopter's estimated velocity and corresponding estimation error when following an 8-shape trajectory for different cases. For (**a**) position for known landmarks and (**b**) position for unknown landmarks, true states (——), estimated states (- - -), and desired states (-·-·-). For (**c**) estimated position error for known landmarks and (**d**) estimated position error for unknown landmarks, estimation error (——), and 3σ boundary (-·-·-).

The results for attitude estimations were astoundingly much better compared to other states estimated for the quadcopter as shown in Figure 10. The average RMSE for estimation of Euler angles during the simulation was just 0.34 deg for even the unknown landmarks case quite comparable to the known landmarks case, which hovered around 0.17 deg.

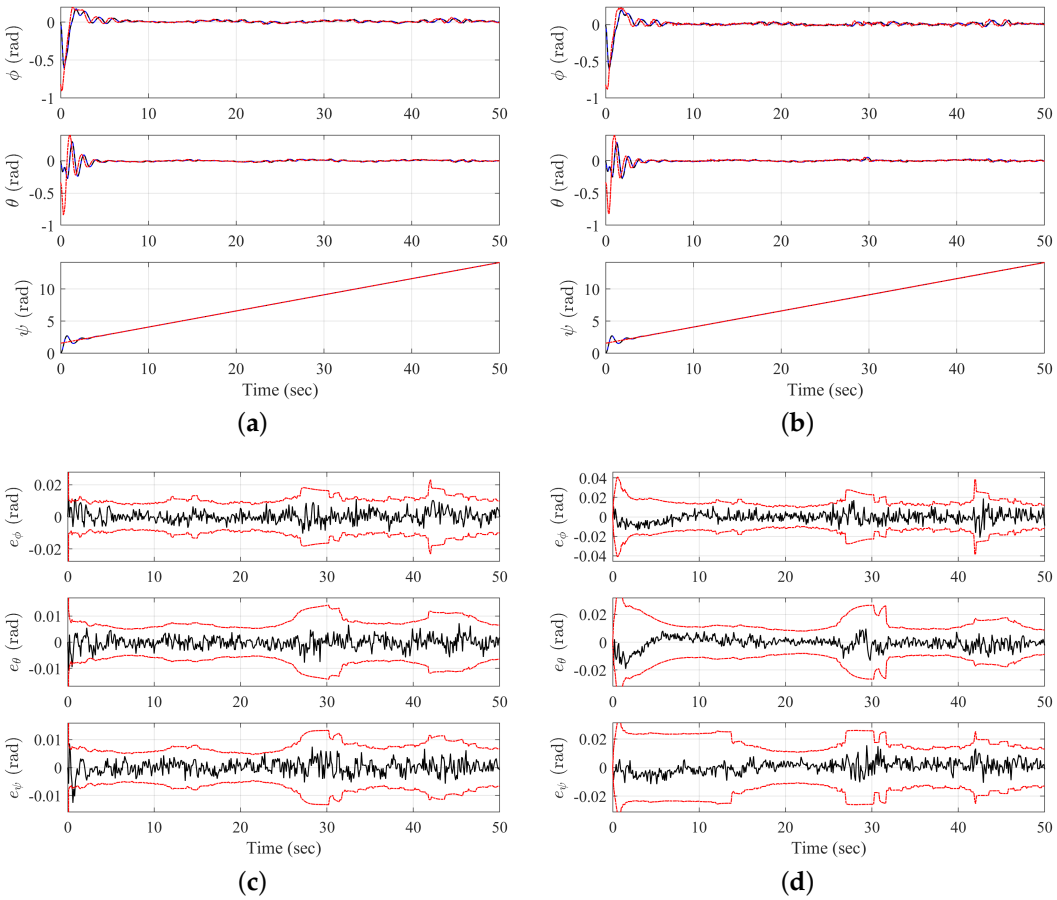

Figure 10. Quadcopter's estimated Euler angles and corresponding estimation error when following an 8-shape trajectory for different cases. For (**a**) attitude for known landmarks and (**b**) attitude for unknown landmarks, true states (——), estimated states (- - -), and desired states (-·-·-). For (**c**) estimated attitude error for known landmarks and (**d**) estimated attitude error for unknown landmarks, estimation error (——), and 3σ boundary (-·-·-).

It should also be noted that the RMSE for each of the estimated states includes the initial high fluctuations before converging to the trajectory. This is mainly a result of the quadcopter's initial position being at the origin, which is further from the initial desired trajectory position. It results in rapid motion of the quadcopter to reach the initial desired position and thus the higher fluctuations. Considering the motion of the quadcopter mainly when it has converged to its desired trajectory, the average RMSE is slightly lower for both the known and unknown landmark scenarios.

3.2. Estimation of Landmarks

The landmarks' inertial position errors are estimated using EKF-SLAM and are depicted in Figure 11. Figure 11a shows that each of the inertial coordinates for each of the landmarks is estimated within the 3σ boundary. It is visible in the figure that some landmarks are initialized after some time because they were sensed at that instant only. Thereafter, with the knowledge of uncertainties for previously identified and estimated landmarks, the uncertainty for newly discovered landmarks decreases over time. Figure 11b shows the final state of estimation error for all the landmarks at the end of the simulation. It can be seen that a few landmarks were never sensed and thus not initialized

for estimation. The average RMSE for the estimated position for landmarks was around 0.03 m.

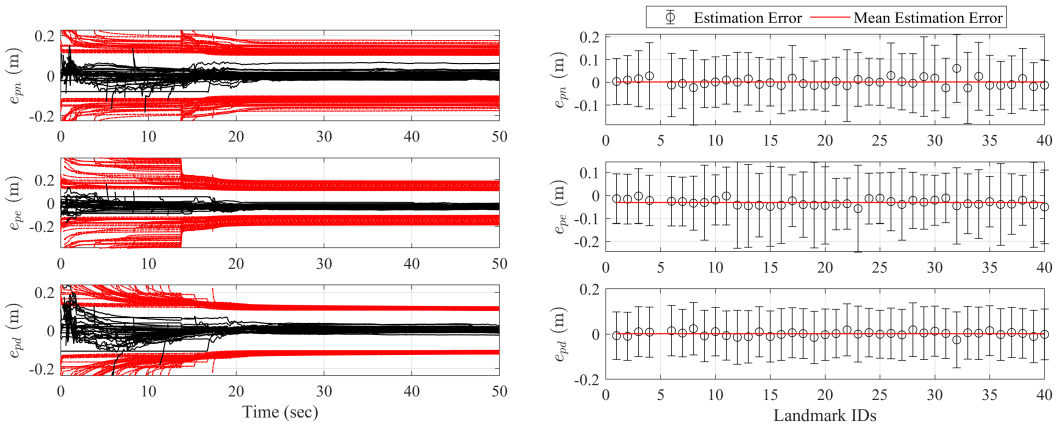

Figure 11. Landmarks state estimation error for an 8-shape trajectory. For 11**a**, estimation error (——), 3σ boundary (- · - ·); each line signifying estimation error for each landmark over time. (**a**) Estimation error over time and (**b**) estimation error at the end.

3.3. Computational Time Analysis for EKF-SLAM

The computational complexity of EKF-SLAM increases quadratically with the number of landmarks [7]. For a different number of landmarks ranging from 5 to 200, the averaged computational time for each of the major steps of EKF-SLAM is displayed in Figure 12. The prediction step of EKF is directly correlated to the number of mapped landmarks as the size of the cross-covariance matrix propagated in Equation (29) increases with it. Thus, the averaged computational time for the prediction step is seen varying quadratically with respect to the number of mapped landmarks in Figure 12a. Likewise, the computation time for the update depends on the number of landmarks that are observed and mapped. The linear trend of averaged computational time for the update considered with respect to the observed mapped landmarks is shown in Figure 12b, as the update step considers observed mapped landmarks. Lastly, the registration time in SLAM is also linearly dependent on the number of newly discovered landmarks that are registered as depicted in Figure 12c. It is evident from the figures that the computational effort of EKF-SLAM certainly increases quadratically with the number of landmarks. However, the main idea for generating these plots is to develop a metric or a strategy to run EKF-SLAM in real-time with a choice of the appropriate number of landmarks at each step, minimally affecting the total performance.

For instance, for better performance of EKF-SLAM, it is ideal not to miss any sensor data. Thus, considering the sensors' sampling rate of 10 Hz, the steps of EKF-SLAM should ideally be complete within 0.1 s. Hence, if for example, the registration step is taking almost 0.1 s, it can be strategically decided to consider a finite number of landmarks using Figure 12c. Similarly, in case the update step itself is taking longer with making updates using all the observed mapped landmarks, only a few of those landmarks can be used to limit the time of operation to less than 0.1 s. Additionally, it is clear from Figure 12 that the update and registration steps take more time than the prediction step. Additionally, only after 200 mapped landmarks does prediction have a significant effect on computation time. Hence, if a mission identifies less than around 200 landmarks, decisions can be based upon choosing an appropriate number of landmarks for update and registration. However, in case the number of mapped landmarks is more, decisions can be made where only a few landmarks at random are taken into consideration during the prediction step. It should be noted the results in Figure 12 will vary among different computers. The results shown were generated using a computer with Intel i7-11800H 2.30 GHz processor, 16 GB RAM, and an NVIDIA T1200 GPU 4 GB. However, these sets of plots will be similar and can be used to

make decisions on the number of landmarks to consider for each step of EKF-SLAM to improve its real-time performance.

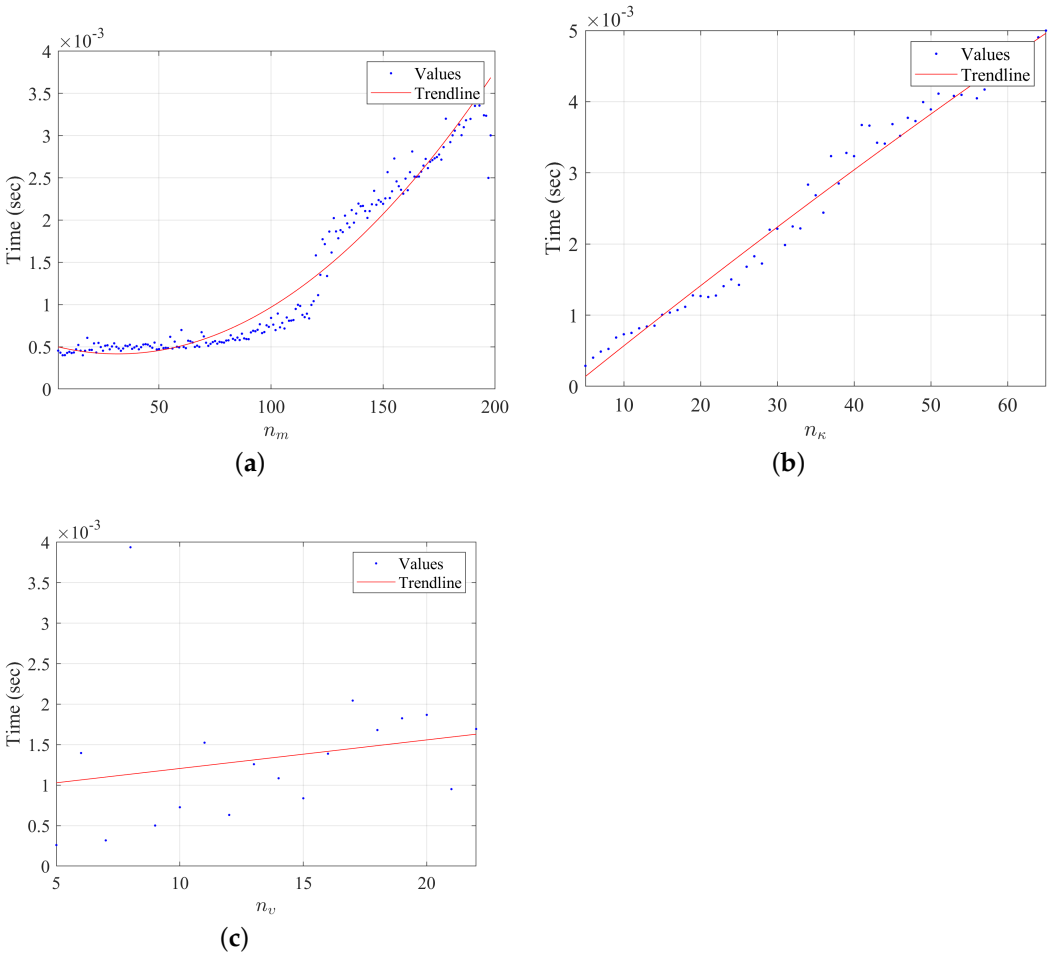

Figure 12. Computational time for different steps of EKF-SLAM with respect to the number of landmarks. (**a**) Computational time of prediction step vs. number of mapped landmarks, (**b**) computational time of update step vs. number of observed mapped landmarks, and (**c**) computational time of registration step vs. number of observed unmapped landmarks.

4. Conclusions

This work proposes an Extended Kalman Filter-based Simultaneous Localization And Mapping (EKF-SLAM) for a quadcopter to follow aggressive trajectories using a Differential Flatness-based Linear Quadratic Regulator (DF-LQR). A strategy to reduce the computational effort of EKF-SLAM using averaged computational time of operation for each step of the process with respect to the number of landmarks is also proposed in the end. To validate the performance of the proposed approach, the quadcopter traversing an 8-shape trajectory is considered for two scenarios of known landmarks and unknown landmarks, and to imitate practical conditions, sensors' constraints, such as limited sensing ranges and field of views, are considered. The estimation errors for the quadcopter states are comparable for both cases despite the lack of information on the landmarks in the unknown landmarks scenario. When following the given trajectory, the estimated states of both the quadcopter and landmarks are accurate enough for DF-LQR to generate the appropriate control signal using the estimated states themselves. The estimated position, velocity, and attitude of the quadcopter are extremely accurate along with the estimation of the landmark positions evident from extremely low root-mean-square errors. The computation time analysis for each step of EKF-SLAM with respect to the number of landmarks shows that the computational time does increase quadratically with the number of landmarks, and

the results are identified to be beneficial in strategically choosing the respective number of landmarks for each step to maximize the use of sensor data and improve performance. Albeit this work validates the proposed approach via numerical simulations, hardware experiments will be conducted to verify its real-time applicability during the quadcopter's aggressive maneuvers in future work.

Author Contributions: Conceptualization, S.R. and S.B.; methodology, S.R., S.B., D.C. and D.K.; software, S.R. and S.B.; validation, S.R., S.B. and D.C.; formal analysis, S.R., S.B., D.C. and D.K.; investigation, S.R. and S.B.; resources, D.K.; data curation, S.R. and S.B.; writing—original draft preparation, S.R. and S.B; writing—review and editing, D.C. and D.K.; visualization, S.R., S.B. and D.C.; supervision, D.K.; project administration, D.K.; funding acquisition, D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]
2. Sadeghzadeh-Nokhodberiz, N.; Can, A.; Stolkin, R.; Montazeri, A. Dynamics-based modified fast simultaneous localization and mapping for unmanned aerial vehicles with joint inertial sensor bias and drift estimation. *IEEE Access* **2021**, *9*, 120247–120260. [CrossRef]
3. Suzuki, T.; Amano, Y.; Hashizume, T. Development of a SIFT based monocular EKF-SLAM algorithm for a small unmanned aerial vehicle. In Proceedings of the SICE Annual Conference 2011, Tokyo, Japan, 13–18 September 2011; pp. 1656–1659.
4. Karam, S.; Nex, F.; Chidura, B.T.; Kerle, N. Microdrone-Based Indoor Mapping with Graph SLAM. *Drones* **2022**, *6*, 352. [CrossRef]
5. Balamurugan, G.; Valarmathi, J.; Naidu, V. Survey on UAV navigation in GPS denied environments. In Proceedings of the 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, India, 3–5 October 2016; pp. 198–204.
6. Smith, R.; Self, M.; Cheeseman, P. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*; Springer: New York, NY, USA, 1990; pp. 167–193.
7. Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping: Part I. *IEEE Robot. Autom. Mag.* **2006**, *13*, 99–110. [CrossRef]
8. Takleh, T.T.O.; Bakar, N.A.; Rahman, S.A.; Hamzah, R.; Aziz, Z. A brief survey on SLAM methods in autonomous vehicle. *Int. J. Eng. Technol.* **2018**, *7*, 38–43. [CrossRef]
9. Cho, Y.; Hwang, J.Y. A study on EKF-SLAM simulation of autonomous flight control of quadcopter. *Int. J. Softw. Eng. Its Appl.* **2015**, *9*, 269–282. [CrossRef]
10. Azizi, A.; Nourisola, H.; Ghiasi, A.R. 3D inertial algorithm of SLAM for using on UAV. In Proceedings of the 2016 4th International Conference on Robotics and Mechatronics (ICROM), Tehran, Iran, 26–28 October 2016; pp. 122–129.
11. Hening, S.; Ippolito, C.A.; Krishnakumar, K.S.; Stepanyan, V.; Teodorescu, M. 3D LiDAR SLAM integration with GPS/INS for UAVs in urban GPS-degraded environments. In Proceedings of the AIAA Information Systems-AIAA Infotech@ Aerospace, Grapevine, TX, USA, 9–13 January 2017; p. 0448.
12. Kim, H.; Kim, D.; Kim, S. Real-time Geospatial Positioning for UAVs in GPS-Denied Environment Using LiDAR Data. In Proceedings of the AIAA Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020; p. 2194.
13. Yu, W.; Zamora, E. Sliding mode three-dimension SLAM with application to quadrotor helicopter. In Proceedings of the 2018 15th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), Mexico City, Mexico, 5–7 September 2018; pp. 1–6.
14. Bagheri, S.; Jafarov, T.; Freidovich, L.; Sepehri, N. Beneficially combining LQR and PID to control longitudinal dynamics of a SmartFly UAV. In Proceedings of the 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 13–15 October 2016; pp. 1–6.
15. Ferrin, J.; Leishman, R.; Beard, R.; McLain, T. Differential flatness based control of a rotorcraft for aggressive maneuvers. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 2688–2693.
16. Paz, L.M.; Tardós, J.D.; Neira, J. Divide and conquer: EKF SLAM in $o(n)$. *IEEE Trans. Robot.* **2008**, *24*, 1107–1120. [CrossRef]
17. Beard, R.W. Quadrotor dynamics and control. *Brigh. Young Univ.* **2008**, *19*, 46–56.
18. Luukkonen, T. Modelling and control of quadcopter. *Indep. Res. Proj. Appl. Math. Espoo* **2011**, *22*, 22.
19. Martins, L.; Cardeira, C.; Oliveira, P. Linear quadratic regulator for trajectory tracking of a quadrotor. *IFAC-PapersOnLine* **2019**, *52*, 176–181. [CrossRef]

20. Rigatos, G.G. Differential flatness theory and flatness-based control. In *Nonlinear Control and Filtering Using Differential Flatness Approaches*; Springer: Cham, Switzerland, 2015; pp. 47–101.
21. Tedrake, R. Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for MIT 6.832. *Work. Draft. Ed.* **2009**, *3*, 91–92.
22. Hespanha, J.P. *Linear Systems Theory*; Princeton University Press: Princeton, NJ, USA, 2018.
23. Saraf, P.; Gupta, M.; Parimi, A.M. A Comparative Study Between a Classical and Optimal Controller for a Quadrotor. In Proceedings of the 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India, 10–13 December 2020; pp. 1–6.
24. Leishman, R.C.; Macdonald, J.C.; Beard, R.W.; McLain, T.W. Quadrotors and accelerometers: State estimation with an improved dynamic model. *IEEE Control. Syst. Mag.* **2014**, *34*, 28–41.
25. Sola, J. Simultaneous localization and mapping with the extended Kalman filter. *Oct* **2014**, *5*, 35.
26. Bailey, T.; Durrant-Whyte, H. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robot. Autom. Mag.* **2006**, *13*, 108–117. [CrossRef]