Sensors **2014**, *14*, 21791-21825; doi:10.3390/s141121791

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

Sampling-Based Real-Time Motion Planning under State Uncertainty for Autonomous Micro-Aerial Vehicles in GPS-Denied Environments

Dachuan Li [1,*], Qing Li [1], Nong Cheng [1,2] and Jingyan Song [1]

[1] Department of Automation, Tsinghua University, Bejing 100084, China;
E-Mails: liqing@tsinghua.edu.cn (Q.L.); ncheng@tsinghua.edu.cn (N.C.);
jysong@tsinghua.edu.cn (J.S.)

[2] National Key Laboratory on Flight Vehicle Control Integrated Technology,
Flight Automatic Control Research Institute, Xi'an 710065, China

* Author to whom correspondence should be addressed; E-Mail: DachuanLi86@gmail.com;
Tel.: +86-151-2000-0153; Fax: +86-010-6279-7972.

External Editor: Felipe Jimenez

Received: 31 July 2014; in revised form: 26 October 2014 / Accepted: 3 November 2014 / Published: 18 November 2014

Abstract: This paper presents a real-time motion planning approach for autonomous vehicles with complex dynamics and state uncertainty. The approach is motivated by the motion planning problem for autonomous vehicles navigating in GPS-denied dynamic environments, which involves non-linear and/or non-holonomic vehicle dynamics, incomplete state estimates, and constraints imposed by uncertain and cluttered environments. To address the above motion planning problem, we propose an extension of the closed-loop rapid belief trees, the closed-loop random belief trees (CL-RBT), which incorporates predictions of the position estimation uncertainty, using a factored form of the covariance provided by the Kalman filter-based estimator. The proposed motion planner operates by incrementally constructing a tree of dynamically feasible trajectories using the closed-loop prediction, while selecting candidate paths with low uncertainty using efficient covariance update and propagation. The algorithm can operate in real-time, continuously providing the controller with feasible paths for execution, enabling the vehicle to account for dynamic and uncertain environments. Simulation results demonstrate that the proposed approach can generate feasible trajectories that reduce the state estimation uncertainty, while handling complex vehicle dynamics and environment constraints.

Keywords: motion planning; micro-aerial vehicles; rapidly exploring random trees (RRT); state estimation uncertainty

1. Introduction

Autonomous micro-aerial vehicles (MAV) are playing an increasingly important role in many civil and military applications. MAVs that are capable of autonomously making decisions and operating can be applied in many tasks scenarios that are not accessible by humans and ground mobile robots, such as indoor exploration and mapping, search and rescue, disaster relief, *etc.* As a result, there has been an increasing interest in developing autonomous MAV systems for indoor navigation. In recent years, many researchers have developed and implemented various kinds of MAV systems demonstrating various degrees of autonomy in performing tasks, such as indoor exploration [1], environment mapping [2] and agile flight [3].

Despite the considerable progress achieved in this domain, there are still many challenges in developing fully autonomous MAV systems. One of the key problems is the motion/path planning for MAVs capable of operating in GPS-denied complex environments. For MAVs performing tasks in such scenarios, the motion planning algorithm must comply with constraints, including complex vehicle dynamics (non-linear and/or non-holonomic dynamics with a high dimensional state space) and environmental constraints (unstructured and uncertain, time-varying operating environments). Particularly, since MAVs typically cannot directly get access to the information of the current state, they must estimate the distribution over the states using measurements from onboard sensors or external aids (GPS [4], external motion capture system [5]). However, GPS is unreliable in urban canyons and completely unavailable in most indoor environments; the external motion capture system is also impractical for such scenarios, since it requires pre-installation of camera arrays in the environment. As a result, the state estimation of MAVs must rely only on onboard sensing capabilities, which is highly constrained in terms of precision and range due to the size and weight constraints of the MAVs. Moreover, the performance of MAV state estimation and localization using exteroceptive sensors (laser rangefinder [6], camera [7] and RGB-D sensor [8]) varies across the environment depending on the distinctive features and perceptual structure of the environments. As a result, the motion planning progress must also integrate the uncertainty of state estimation to ensure reliability and robustness to imperfect and noisy state estimates caused by limited sensing capability. These challenges require that the motion/path planner must be able to generate feasible paths that satisfy the constraints imposed by uncertain and unstructured environments, as well as complex vehicle dynamics, while ensuring measurement gathering along the path to reduce state estimation uncertainty.

In this paper, we propose a real-time motion planning strategy (closed-loop random belief trees, CL-RBT) for autonomous vehicles with complex dynamics and in the presence of state estimation uncertainty, while handling the constraints imposed by the uncertain and dynamic operating environments. The proposed motion planning strategy is built upon the randomized sampling-based motion framework, the real-time closed-loop rapidly exploring random trees (CL-RRT) [9], which operates by sampling in the input space of the controller and generates trajectories through forward closed-loop simulation using

the vehicle dynamics model and path-tracking controller. This allows the motion planner to easily account for non-linear and non-holonomic vehicle dynamics and dynamic uncertain environments. We extend the general CL-RRT to handle the state uncertainty by incorporating the prediction of posterior state estimation uncertainty in the motion planning framework. The uncertainty of state estimation is characterized using a factored form of the covariance [10] provided by the Kalman filter class-based estimator. This factored form enables efficient covariance propagation in that it combines the update of covariance along a path from multi-step observations into a single linear step, and the posterior covariance of a certain path after adding new nodes can also be updated online. The overall motion planning operates by incrementally constructing a tree of dynamically feasible trajectories in real time, while continuously selecting and executing trajectories that are a tradeoff between minimizing path cost and reducing localization uncertainty. We validated the proposed algorithm in various illustrative scenarios of a simulated quadrotor MAV with non-linear dynamics and limited sensing in enclosed and unstructured GPS-denied environments. Simulation results demonstrate that the CL-RBT can efficiently generate dynamically feasible trajectories that ensure state estimation accuracy, while preserving the property of the CL-RRT framework. The CL-RBT can handle complex vehicle dynamics and state uncertainty, enabling the MAV to autonomously navigate in uncertain, unstructured and GPS-denied environments.

The paper is organized as follows: Section 2 reviews related work on motion planning under uncertainty. Section 3 presents the formulation of the motion planning problem. Section 4 provides the covariance propagation approach. A detailed description of the CL-RBT is presented in Section 5, followed by the simulation results and analysis in Section 6. Finally, the paper is concluded in Section 7, with a discussion on future work.

2. Related Work

Conventionally, motion planning approaches under state uncertainty are typically formulated as a partially observable Markov decision process (POMDP) problem [11], which provides the most general mathematical framework for solving the planning problem with partial observability. While POMDP has been applied to low-dimensional and small-scale problems [12], most POMDP-based approaches rely on discretizing the state space, making it computationally intractable for realistic applications. Recently, many approaches have been proposed to address the problem of scalability using approximation and iterative methods. Van den Berg *et al.* [13] proposed an iterative motion planning approach to solve the continuous POMDP problem, using a belief space variant of the iterative LQG (linear-quadratic Gaussian) method. Similarly, Bai *et al.* [14] also formulated the problem as a continuous POMDP and solved the problem with a Monte Carlo value iteration method. However, these approaches are still less effective at addressing problems with complex vehicle dynamics and motion planning with large-scale, high dimensional state space or configuration space.

In contrast, sampling-based motion planning strategies have been widely acknowledged as effective approaches for solving motion planning problems with high dimensional configuration space and complex vehicle dynamics. In particular, the PRM (probabilistic roadmap) [15] and RRT (rapidly exploring random trees) [16] and their variants are the most widely applied sampling-based approaches, and they have been successfully applied in a number of applications. More recently, the closed-loop RRT [9,17,18] extends

the conventional RRT by incorporating forward prediction using closed-loop dynamics model with controller. The CL-RRT can generate more feasible paths that account for complex vehicle dynamics, and it can be implemented in real-time to handle dynamic, uncertain environments. Several approaches have been proposed to incorporate state uncertainty in the sampling-based motion planning framework. Van den Berg *et al.* [19] proposed the LQG-MP strategy by integrating the *a priori* distribution over state estimates into the standard RRT framework. However, the LQG-MP algorithm is intractable for real-time applications, due to its high computation cost. In [20], Bry *et al.*, combined an LQG-based covariance pruning technique with the RRT* framework for planning in belief space. Alternative to RRT-based approaches, the belief roadmap (BRM) extends the PRM by integrating the predictions of state estimation uncertainty using an EKF (extended Kalman filter) estimator and performs belief planning by searching paths with the lowest uncertainty from the roadmap. The BRM was later extended to use the UKF estimator and heuristic sampling strategy [21], and it was further implemented on a quadrotor and proven successful for indoor flight tests [22]. However, the BRM assumes that the vehicle is fully controllable and treats the problem as a simple kinematic motion planning problem, e.g., it does not consider complex vehicle dynamics and the feasibility of the generated paths. As a result, it is not yet well suited to vehicles with complex dynamic constraints and dynamic environments. In addition, the BRM operates by planning on pre-constructed static graphs, and therefore, it is not a real-time planning algorithm in essence. Recent research has also considered another similar kind of planning problem with uncertainty, where the planner must generate paths that maximize information collection and reduce the uncertainty on the location estimates of features or targets in the environment. The information-rich RRT (IRRT) [23] addresses this path planning problem by incorporating the qualification of information gain into the CL-RRT framework, using the Fisher information matrix. This kind of problem can be regarded as the inverse problem of planning with uncertainty on the vehicle's own state.

3. Problem Formulation

3.1. Motion Planning under State Uncertainty

Consider a vehicle with non-linear dynamics that operates in an environment without external positioning systems. The vehicle typically does not have direct access to the accurate information of its current state, but instead obtains the estimates of the state using observations derived from the measurements of onboard sensors, which are typically noisy and incomplete. The stochastic dynamics and observation model of the system can be given by the following discrete time state-transition form:

$$\begin{aligned} \mathbf{x}_{t+1} &= f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) \qquad \mathbf{w}_t \sim N(0, \mathbf{Q}) \\ \mathbf{z}_t &= f(\mathbf{x}_t, \mathbf{v}_t) \qquad \mathbf{v}_t \sim N(0, \mathbf{R}) \end{aligned} \tag{1}$$

where $\mathbf{x} \in X, \mathbf{u} \in U$ and $\mathbf{z} \in Z$ are the vehicle's state, control input and observation, respectively. $\mathbf{w}_t$ and $\mathbf{v}_t$ denote the process disturbance and measurement noise, and both can be formulated as Gaussian noise with zero mean and covariance (**Q** and **R**, respectively). Assuming that all probability distributions are Gaussian, given the previous observation ($\mathbf{z}_{1:t}$) and control inputs ($\mathbf{u}_{1:t}$), the state is typically estimated using a Bayesian filter, providing the distribution of the vehicle's state (or the belief of the state):

$$p(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$$

which is characterized by a mean state μ_t and a covariance $\Sigma_t(b(\mathbf{x}_t) = (\mu_t, \Sigma_t))$. The mean μ_t provided by the Bayesian filter is a optimal estimation of the vehicle's state distribution, and it can be used for control and decision making, while the mean Σ_t characterizes the confidence (or uncertainty) of state estimation.

Denoting X_{free} as the subset of all collision-free states, given the initial state $\mathbf{x}_0 \in X_{free}$ and the goal region $X_{goal} \subset X_{free}$, as well as the partial knowledge of the environment, then the primary objective of motion planning is to find the control policy $\mathbf{u}_{0:T} = \pi[b(\mathbf{x}_{0:T})]$ and corresponding sequence of states $\mathbf{x}_{0:T}$, such that the vehicle reaches the goal region in a finite time horizon by applying the control policy:

$$\mathbf{x}_T \in X_{goal,} t \in (0, t_f], t_f \in (0, \infty)$$

meanwhile minimizing the following objective function:

$$J(b(\mathbf{x}_t)) = E_{b(\mathbf{x}_{goal})|b(\mathbf{x}_t),\mathbf{u}_{0:T}}[C(\mathbf{x}_t \mid \mathbf{x}_{goal})] + \sum_{t=0}^{T} C(\mathbf{x}_t, \mathbf{u}_t)$$

where $C(\mathbf{x}_t|\mathbf{x}_{goal})$ is the expected cost function from $\mathbf{x}_t$ to $\mathbf{x}_{goal}$; note that it takes the expectation form, since the measurement and state are both probabilistic, and $C(\mathbf{x}_t, \mathbf{u}_t)$ is the cost by applying control $\mathbf{u}_t$. The generated trajectory must also satisfy the constraints imposed by the vehicle dynamics Equation (1) and environments (*i.e.*, collision avoidance, $\mathbf{x}_{0:T} \in X_{free}$). Moreover, when the vehicle does not have accurate knowledge of the estimate of its state, the motion planner must take the uncertainty of the state estimate along the generated paths into account, thus the motion planning problem into the belief planning problem (*i.e.*, planning in belief space [20]). By integrating the information from the mean and covariance of the belief, the motion planning algorithm must choose actions and beliefs, such that the posterior covariance at the end of the trajectory is minimized, yielding trajectories that achieve a trade-off between low path cost and high confidence in the state estimation.

3.2. State Estimation of Gaussian Systems

Consider a system with the dynamic and observation model in the form of Equation (1), one of the most common and robust methods for estimating the distribution over its state is the Bayesian filtering [10]. Given knowledge of the prior control input $\mathbf{u}_t$ and sensor measurement $\mathbf{z}_{t+1}$, the belief of the state $b(\mathbf{x}_{t+1})$ after a sequence of control and observation can be estimated as:

$$b(\mathbf{x}_{t+1}) = p(\mathbf{x}_{t+1} \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \lambda p(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) \int p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) b(\mathbf{x}_t) d\mathbf{x}_t \tag{2}$$

where λ is a normalization factor.

Assuming the state and observation transition function (f, h) are linear with state and observation, which are both Gaussian distributions; the Bayesian filtering can be implemented as the Kalman filter [24], while the extended Kalman filter (EKF) [25] is typically applied for estimating the state distribution of the system with nonlinear state and observation transition functions. For a system model of the form of Equation (1), the EKF state estimation can be divided into the process step and measurement update step. Denoting the distribution of the state $b(\mathbf{x}_t) = N(\hat{\mathbf{x}}_t, \Sigma_t)$, the process step first predicts the state using the control and state of the previous step, which is given as:

$$\begin{aligned} \overline{\hat{\mathbf{x}}}_t &= g(\hat{\mathbf{x}}_{t-1}, \mathbf{u}_t) \\ \overline{\Sigma}_t &= G_t \Sigma_{t-1} G_t^T + V_t Q_t V_t^T \end{aligned} \tag{3}$$

where G_t is the Jacobian of g with respect to the state, and V_t is the Jacobian of g with respect to $\mathbf{w}_t$. Then, the measurement step adjusts the estimate and covariance by incorporating the information from new observations:

$$\begin{aligned} \hat{\mathbf{x}}_t &= \bar{\hat{\mathbf{x}}}_t + K_t(\mathbf{z}_t - H_t\bar{\hat{\mathbf{x}}}_t) \\ \Sigma_t &= \bar{\Sigma}_t - K_t H_t \bar{\Sigma}_t \end{aligned} \tag{4}$$

where H_t denotes the Jacobian of h with respect to the state, and K_t is the Kalman gain, which is updated by:

$$K_t = \bar{\Sigma}_t H_t^T (R_t + H_t \bar{\Sigma}_t H_t^T)^{-1}$$

Typically, the control and decision making are based on the mean $\hat{\mathbf{x}}_t$ of the estimated distribution provided by the EKF, and the covariance, while the covariance Σ_t captures the uncertainty of the state estimate using the sensor measurements. Therefore, the uncertainty of the state estimate along the trajectory from the path planner can be evaluated using the norm of the covariance, which will be discussed in the next section.

4. Linear Covariance Propagation

In order to evaluate the state uncertainty resulting from a specific planned trajectory, the most common approach for a system with the EKF estimator is to compute the posterior covariance of the ending belief $b(\mathbf{x}_t)$ from the sequence of actions and measurements along the trajectory. However, the propagation of the posterior covariance requires multiple iterative calculations of the EKF process and measurement updating from the initial belief according to Equation (4), leading to a heavy computational cost. This is even worse when the initial belief is modified, since the posterior covariance must be re-computed using the entire EKF updates from the new initial belief. In particular, this fact has a more significant effect on a sampling-based motion planner, since it generally constructs a graph and tree of multiple trajectories, thus different paths can result in different posterior covariance to the same node, this requires that the motion planner must perform the propagation process for each covariance.

To reduce the computational cost of the covariance propagation, we rely on previous results on the factorization of the covariance [10], which allows the propagation of posterior covariance to be propagated in a single linear update step instead of multiple non-linear updates for the EKF filter.

Following Theorem 1 in [10], the factorization of covariance is given by:

$$\Sigma_t = \Lambda_t \Pi_t^{-1}$$

where Λ_t and Π_t can be calculated as linear functions of Λ_{t-1} and Π_{t-1}, using the EKF process and measurement update step.

The factorization of the covariance matrix can be proved using the following matrix inversion lemma:

Lemma 1. For matrices $M_1, M_2, M_3 \in \mathrm{R}^{n\times n}$, we have:

$$(M_1 + M_2 M_3^{-1})^{-1} = M_3(M_2 + M_1 M_3)^{-1} \tag{5}$$

Denote the initial state covariance as Σ_0, and Σ_0 can be factored as:

$$\Sigma_0 = \Sigma_0 \mathbf{I}^{-1} \tag{6}$$

Given $\Sigma_{t-1} = \Lambda_{t-1}\Pi_{t-1}^{-1}$ and denoting $S_t = V_t Q_t V_t^T$, the process update of EKF (Equation (3)) can be written as:

$$\bar{\Sigma}_t = G_t\Sigma_{t-1}G_t^T + S_t = G_t\Lambda_{t-1}\Pi_{t-1}^{-1}G_t^T + S_t = G_t\Lambda_{t-1}(G_t^{-T}\Pi_{t-1})^{-1} + S_t$$

Following Equation (5), we have:

$$\bar{\Sigma}_t = \left((G_t^{-T}\Pi_{t-1})(G_t\Lambda_{t-1} + S_t G_t^{-T}\Pi_{t-1})^{-1}\right)^{-1} = (\bar{E}_t\bar{F}_t^{-1})^{-1} = \bar{F}_t\bar{E}_t^{-1} \tag{7}$$

where $\bar{E}_t = G_t^{-T}\Pi_{t-1}$ and $\bar{F}_t = G_t\Lambda_{t-1} + S_t G_t^{-T}\Pi_{t-1}$,

For computation considerations, the covariance update of the measurement update step can be given by the information form [11]:

$$\begin{aligned} \bar{\Omega}_t &= \bar{\Sigma}_t^{-1} = (G_t\Omega_{t-1}^{-1}G_t^T + T_t)^{-1} \\ \Omega_t &= \bar{\Omega}_t + H_t^T R_t^{-1} H_t \end{aligned} \tag{8}$$

Denoting $N_t = H_t^T R_t^{-1} H_t$, Equation (8) can be rewritten as $\Omega_t = \bar{\Omega}_t + N_t$. Therefore, from Equations (7) and (8), we can write:

$$\Sigma_t = \Omega_t^{-1} = (\bar{\Sigma}_t^{-1} + N_t)^{-1} = (\bar{E}_t\bar{F}_t^{-1} + N_t)^{-1}$$

Following Equation (5),

$$\Sigma_t = \bar{F}_t(\bar{E}_t + N_t\bar{F}_t)^{-1} = \Lambda_t\Pi_t^{-1}$$

where:

$$\begin{aligned} \Lambda_t &= \bar{F}_t = G_t\Lambda_{t-1} + S_t G_t^{-T}\Pi_{t-1} \\ \Pi_t &= \bar{E}_t + N_t\bar{F}_t = G_t^{-T}\Pi_{t-1} + N_t(G_t\Lambda_{t-1} + S_t G_t^{-T}\Pi_{t-1}) \\ &= N_t G_t\Lambda_{t-1} + (I + N_t S_t)G_t^{-T}\Pi_{t-1} \end{aligned} \tag{9}$$

As can be seen from Equation (9), Λ_t, Π_t are both linear functions of Λ_{t-1}, Π_{t-1}.

Denoting Ψ as the stacked block matrix of Λ and Π:

$$\Psi = \begin{bmatrix} \Lambda \\ \Pi \end{bmatrix} \tag{10}$$

Equation (10) can be rewritten as:

$$\Psi_t = \begin{bmatrix} \Lambda_t \\ \Pi_t \end{bmatrix} = \begin{bmatrix} 0 & I \\ I & N \end{bmatrix}_t \begin{bmatrix} 0 & G^{-T} \\ G & SG^{-T} \end{bmatrix}_t \begin{bmatrix} \Lambda_{t-1} \\ \Pi_{t-1} \end{bmatrix} = \varsigma_t\Psi_{t-1} \tag{11}$$

where:

$$\varsigma_t = \begin{bmatrix} 0 & I \\ I & N \end{bmatrix}_t \begin{bmatrix} 0 & G^{-T} \\ G & SG^{-T} \end{bmatrix}_t = \begin{bmatrix} G & SG^{-T} \\ NG & G^{-T} + NSG^{-T} \end{bmatrix}_t \tag{12}$$

is the one-step transfer function matrix. Therefore, the covariance at t can be recovered using the factors of Ψ_t:

$$\Sigma_t = \Lambda_t\Pi_t^{-1} \tag{13}$$

Using the above factorization and the linear transfer function, the non-linear Kalman update process of the covariance can be transformed into a linear propagation step of the covariance factors, and the multiple updates can be combined into a single transfer step. This allows for efficient covariance propagation and uncertainty prediction for the sampling-based motion planning strategy: The posterior

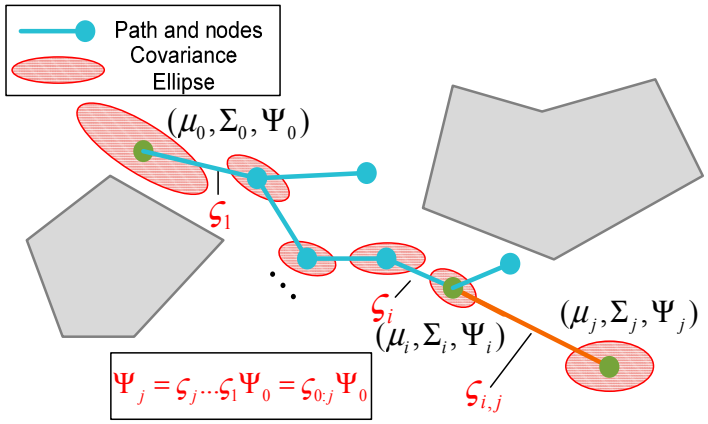
covariance Σ_T resulting from any initial belief (μ_t, Σ_t) along the trajectory can be recovered by multiplying multiple transfer functions:

$$\Psi_T = \varsigma_T \cdots \varsigma_{t+1} \Psi_t = \varsigma_{T-t} \Psi_t \tag{14}$$

and in a sampling-based path planner that represents paths in the form of trees and nodes, for any arbitrary node n_j in the trajectory, the posterior covariance of the state at the node can be propagated from the predecessor node $j–1$ along the path in one efficient step using the transfer function $\varsigma_{0:j}$ (Figure 1):

$$\Psi_j = \varsigma_j \cdots \varsigma_1 \Psi_0 = \varsigma_{0:j} \Psi_0 \tag{15}$$

Figure 1. Linear covariance prediction procedure using the transfer function.



Taking advantage of the above linear covariance propagation approach, the state estimation covariance of one point n_i (node or time step) can be predicted using the covariance of its adjacent point, as well as the EKF Jacobian matrix set associated with the trajectory between n_{i-1} and n_i; this procedure can be given as follows:

Step 1: Extract the covariance factors Λ_{i-1} and Π_{i-1} from block matrix Ψ_{i-1} of n_{i-1}.
Step 2: Compute the EKF matrix set (S_i, G_i and N_i) based on the state transition model and observation model, as well as the predicted state and simulated observation (Equation (1)).
Step 3: Compute the transfer function ς_i using S_i, G_i and N_i, according to Equation (12).
Step 4: Propagate the block matrix Ψ_i of n_i based on ς_i and Ψ_{i-1}, according to Equation (11), then extract the covariance factors Λ_i and Π_i from block matrix Ψ_i.
Step 5: Recover the covariance Σ_i of n_i using Λ_i and Π_i: $\Sigma_i = \Lambda_i \Pi_i^{-1}$.

This covariance propagation procedure allows the path planning algorithm to incorporate the covariance-based qualification of state estimation uncertainty into the planning progress in an efficient way. In the CL-RBT algorithm, which is described in detail in the following section, the state estimation uncertainty is qualified using the trace of the covariance:

$$J(\Sigma_i) = tr(\Sigma_i) \tag{16}$$

5. Closed-Loop Random Belief Trees Algorithm (CL-RBT)

5.1. Data Structure

Similar to the conventional RRT, the CL-RBT operates by constructing a tree of trajectories in the state space (more precisely, the belief space). The tree is defined by nodes and edges that connect different nodes. The primary information stored in a node n consists of the mean of the state, uncertainty qualification, the path cost and the parent node:

$$n_i \triangleq \{\mu_i, \Sigma_i, \Psi_i, J_i(\Sigma), C(n_i), n_i.\text{parent}\}$$

where μ_i and Σ_i represent the mean and the covariance of the state distribution, respectively. Ψ_i is the factored form of the covariance. $J_i(\Sigma)$ qualifies the uncertainty along the trajectory form the root node to node n_i, which can be calculated using the covariance propagation process described in Section 4: $J_i(\Sigma) = tr(\Sigma_i)$; $C(n_i)$ stores the data of the path execution cost associated with node n_i, which can be split into two parts:

$$C_i = \omega_1 c(n_i \mid n_{root}) + \omega_2 \hat{c}(x_{goal} \mid n_i)$$

where $c(n_i|n_{root})$ denotes the accumulated execution cost resulting from following the trajectory form tree root n_{root} to n_i, and $\hat{c}(x_{goal} \mid n_i)$ is the estimated cost from n_i to the goal state x_{goal} (cost-to-go). $\omega_1, \omega_2 \in [0,1]$ are weight factors. n_i.parent represents the index for the parent node of n_i.

Every two neighbor nodes (n_i, n_j) in the tree is connected by an edge $e(n_i, n_j)$, which represents the closed-loop prediction trajectory $\{x_i...x_j\}$ and control policy to steer the vehicle from x_i to x_j.

5.2. Tree Expansion

The CL-RBT motion planning strategy is built upon the standard closed-loop RRT (CL-RRT) originally proposed by [9] and later extended in [17,18]. Similar to the CL-RRT, our CL-RBT algorithm can be split into two primary processes: a tree expansion process that explores the environment by growing the tree and a path selection and execution loop that selects and executes the optimal portion of the tree. Details of the tree expansion process are described in Algorithm 1.

Unlike the standard RRT that generates candidate control inputs randomly or from a look-up table, the tree expansion process of the CL-RBT predicts the feasible trajectory by incorporating a closed-loop system model consisting of the trajectory-following controller and the vehicle dynamics model (Figure 1). Once the n_{near} is determined, the planner generates a reference input $\hat{r}$ (Line 7) using n_{near} and x_{samp}; then $\hat{r}$ is sent to the closed-loop system consisting of the controller and the vehicle dynamics model: the controller generates the control input $\hat{u}$ (Line 8), which is then sent to the vehicle dynamics to predict the state output $\hat{x}$ by forward simulation (Line 9). The predicted trajectory is then checked against environmental constraints (e.g., collision avoidance) to ensure feasibility.

The CL-RBT algorithm extends the CL-RRT tree expansion by incorporating the prediction of the state (position) uncertainty in growing the tree, using the EKF estimator and the covariance propagation approach (Section 4). First, a node $x_{samp} \in X_{free}$ is generated by random sampling in the free space (Line 1). After that, the prediction of state estimation uncertainty is incorporated as heuristic information

in the nearest-node selection: The nearest node n_{near} to x_{samp} by a certain metric is determined from the current tree using the following hybrid heuristic information:

$$i^* = \arg\min_i \left(\lambda_1 c(n_i \mid n_{root}) + \lambda_2 \hat{c}(x_{samp} \mid n_i) + \lambda_3 \frac{x_{samp}.\hat{J}_{samp} - n_i.J_i}{n_i.J} \right) \quad (17)$$

where λ_1, λ_2, λ_3 are weighting factors for each item. The above heuristic information consists of three components: the exploration heuristic, the optimization heuristic and the uncertainty heuristic.

Algorithm 1. Closed-loop random belief tree: tree expansion.

1	Take a sample x in the reference space
2	Find the nearest node set $N_{near} = \{n_{near(i)}\}$, $(i \geq 1)$ of x_{samp} from tree T, using the hybrid heuristic consisting of the path cost and uncertainty qualification
3	**for** each node n_{near} in the nearest node set N_{near}
4	$k \leftarrow 0$
5	$\hat{x}(t+k) \leftarrow$ final state of N_{near}
6	**while** $\hat{x}(t+k) \in \chi_{free}(t+k)$ and $\hat{x}(t+k)$ has not reached x_{samp} **do**
7	Generate reference input $\hat{r}(t+k)$ from n_{near} to x_{samp}
8	Generate control input $\hat{u}(t+k)$ from feedback control law
9	Simulate $\hat{x}(t+k+1)$ from state propagation model
10	$k \leftarrow k+1$
11	**end while**
12	Generate the predicted trajectory $\bar{x}(t)$, $t \in [t_0, t_1]$
13	Split the predicted trajectory $\bar{x}(t)$ and add intermediate nodes n_i
14	**if** $\bar{x}(t) \in X^{free}$, $\forall t \in [t_0, t_1]$ **then**
15	Add x_{sample} and all intermediate nodes to T_t, **break**
16	**else if** all intermediate nodes are feasible
17	Add intermediate nodes to T_t, **break**
18	**end if**
19	**end for**
20	**for** each newly added feasible node n **do**
21	Update path execution cost estimates for n (**Algorithm 2**)
22	simulate the observations $\bar{\mathbf{z}}(t)$ and transfer function along the feasible trajectory between n and n.parent
23	Update the matrix Ψ_n and posterior covariance Σ_n and uncertainty cost $J(\Sigma)$ of n
24	Update the total cost of n
25	**end for**

The exploration heuristic $\hat{c}(x_{samp} \mid n_i)$ denotes the estimated cost by connecting n_i to the sample x_{samp}, which enables the path planning algorithm to focus on adding new nodes to the tree and quickly exploring the environment.

The optimization heuristic $c(n_i|n_{root})$ denotes the accumulated cost of the path from the root node n_{root} to the candidate node n_i. The purpose of using this optimization heuristic is to bias the tree growth towards paths that reduce the overall accumulated cost.

In order to incorporate the state estimation uncertainty factor into the path planning progress, the CL-RBT adds the uncertainty heuristic in the nearest node selection of the tree expansion. This uncertainty heuristic enables the motion planner to select the node that leads to the lowest posterior state estimation uncertainty if it is connected to x_{samp}, using the predicted state covariance based on the simulated closed-loop state output and observations along the trajectory between candidate nodes and x_{samp}. This is realized by qualifying the posterior uncertainty along the path from n_i and x_{samp} using the linear covariance propagation described in Section 7: given the covariance Σ_i and the associated factored matrix Ψ_i of n_i, the observations $\mathbf{z}_{i:samp}$ and state output $\hat{\mathbf{x}}_{i:samp}$ are predicted first based on the closed-loop system model. After that, the matrix set $S_{i:samp}$, $G_{i:samp}$, $N_{i:samp}$ and the transfer function $\varsigma_{i:samp}$ along the path are approximated. Therefore, the posterior covariance $\hat{\Sigma}_{samp}$ of x_{samp} resulting from moving the MAV along the path between n_i and x_{samp} can be propagated by:

$$\hat{\Psi}_{samp} = \varsigma_{i:samp}\Psi_i = \begin{bmatrix} 0 & I \\ I & N \end{bmatrix}_{i:samp} \begin{bmatrix} 0 & G^{-T} \\ G & SG^{-T} \end{bmatrix}_{i:samp} \begin{bmatrix} \Lambda_i \\ \Pi_i \end{bmatrix} = \begin{bmatrix} \hat{\Lambda}_{samp} \\ \hat{\Pi}_{samp} \end{bmatrix}$$

$$\hat{\Sigma}_{samp} = \hat{\Lambda}_{samp}\hat{\Pi}_{samp}^{-1}$$

hence, the resulting uncertainty cost of x_{samp} can be given by $x_{samp}.\hat{J}_{samp} = tr(\hat{\Sigma}_{samp})$.

The tree expansion process may yield one or more candidate feasible nodes and corresponding trajectories. For each candidate node n and trajectory, the planner calculates the path cost (Line 21), simulates the observation along the trajectory (Line 22) and, then, further computes the posterior covariance at n, as well as the uncertainty qualification of n (Line 23). After updating the total cost, the algorithm finally adds the feasible nodes to the current tree T (Line 24).

5.3. Path Cost Evaluation

As aforementioned in Algorithm 1, once a feasible node is identified and added to the tree, the CL-RRT algorithm evaluates and updates its associated cost (Lines 21–24). In order to address the multiple requirements of operating in complex, GPS-denied environments, the CL-RBT adopts multiple cost metrics that incorporate various factors into these tasks. In CL-RBT, the cost of each node can be divided into two primary categories: the path execution cost and uncertainty cost.

The path execution cost denotes the cost of moving the MAV along the trajectory from the root node n_{root} to the goal region via the specific node n_i, and it can be given as the following form:

$$C_p(n_i) = \int_{t_0}^{t_i} c[x(t), u(t)]dt$$

where $[t_0, t_i]$ is the time interval of the corresponding path (node sequence) $\{n_{root} \ldots n_i\}$, and $c[x(t), u(t)]$ is the cost metric of the node trajectory (*i.e.*, Euclidian distance, Dubins distance, *etc.*). In the CL-RBT, the path execution cost consists of two primary components: the accumulated path execution cost and the cost-to-go:

$$C_p(n_i) = C(n_i \mid n_{root}) + C_{\text{CostToGo}}$$

where $C(n_i|n_{root})$ represents the accumulated execution cost resulting from following the trajectory from n_{root} to n_i. There are two types of cost estimates of the cost-to-go for each node: a lower bound C_{LB} and an upper bound C_{UB}. The lower bound cost-to-go can be given as the distance metric between the MAV's state at the node and the goal state:

$$C_{LB}(n_i) = C(x_{goal} \mid n_i)$$

The update of the upper bound cost-to-go can be described as follows: each time a new feasible node n_i is added to the tree, the CL-RBT algorithm attempts to connect n_i to the goal by predicting the closed-loop trajectory between n_i and x_{goal}. If the trajectory is feasible, the upper bound cost-to-go of n_i can be given as the cost associated with this trajectory, otherwise the upper bound cost-to-go of n_i is set to ∞:

$$C_{UB} = \begin{cases} \int_{n_j}^{x_{goal}} c(x(t),u(t)) & : \text{ feasible trajectory between } n_j, x_{goal} \text{ exists} \\ \infty & : \text{ no feasible trajectory between } n_j, x_{goal} \end{cases}$$

After that, the CL-RBT propagates the paths back towards the root node to update the upper bound cost-to-go of n_i's affected ancestor nodes: the old upper bound cost-to-go of the parent node is compared with the cost following the newly generated feasible trajectory. If the latter is smaller, the upper bound cost-to-go of the parent node is updated; otherwise, the propagation procedure stops, since there exists a sub-path of the parent with a lower cost-to-go. This update procedure repeats until the current root node is reached. The details of the cost update procedure are shown in Algorithm 2.

Algorithm 2. Closed-loop random belief tree: cost update.

1	$n_i.C_{LB} \leftarrow c(x_{goal} \mid n_i)$
2	Compute the accumulated cost of the trajectory between n_{root} and $n_i : C(n_i \mid n_{root})$
3	Generate the trajectory from n_i to goal state $\bar{x}(x_{goal} \mid n_i)$, using the closed-loop system model
4	Check the feasibility of the closed-loop trajectory $\bar{x}(x_{goal} \mid n_i)$
5	**if** $\bar{x}(x_{goal} \mid n_i)$ is feasible **then**
6	TrajctoryToGoal $\leftarrow$ **True**
7	$n_i.C_{UB} \leftarrow C(\bar{x}(x_{goal} \mid n_i))$ (the path cost of $\bar{x}(x_{goal} \mid n_i)$)
8	$n_k \leftarrow n_i$
9	**while** $n_k \neq n_{root}$ **and** $n_k.\text{parent}.C_{UB} > n_k.C_{UB} + C(n_k.\text{parent} \mid n_k)$
10	$n_k.\text{parent}.C_{UB} \leftarrow n_k.C_{UB} + C(n_k.\text{parent} \mid n_k)$
11	$n_k \leftarrow n_k.\text{parent}$
12	**end while**
13	$n_i.C_{\text{CostToGo}} \leftarrow n_i.C_{UB}$
14	**else**
15	TrajctoryToGoal $\leftarrow$ **False**
16	$n_i.C_{UB} \leftarrow \infty$
16	$n_i.C_{\text{CostToGo}} \leftarrow n.C_{LB}$
17	**end if**
18	$n_i.C \leftarrow C(n_i \mid n_{root}) + n_i.C_{\text{CostToGo}}$
19	**return** TrajctoryToGoal

Once a feasible trajectory to the goal is identified, the upper bound cost-to-go is used as the estimated cost-to-go of n_i; otherwise, the lower bound cost-to-go is used. Therefore, the overall path execution cost of node n_i can be given as:

$$C(n_i) = C(n_i \mid n_{root}) + C_{\text{CostToGo}} = C(n_i \mid n_{root}) + \begin{cases} C_{UB} & : \text{TrajectoryToGoal = True} \\ C_{LB} & : \text{TrajectoryToGoal = False} \end{cases} \quad (18)$$

(Note that the cost update procedure also identifies feasible paths that connect the goal to the current tree. This enables the CL-RRT to quickly find paths that reach the goal. As the tree grows, more paths to the goal can be found).

The CL-RBT incorporates the qualification of state estimation uncertainty in the cost function of nodes, such that the generated paths ensure accurate state estimation. As discussed previously, the state estimation uncertainty is evaluated using the uncertainty cost, e.g., the trace of the predicted covariance: $J(\mathrm{n}_i) = tr(\Sigma_{n_i})$. As shown in Algorithm 1, once a feasible node is added to the tree and its path execution cost is updated, the CL-RBT predicts its posterior state covariance through the update steps described in Section 4 and computes its uncertainty cost (Algorithm 1, Lines 22, 23). Taking the path execution cost and uncertainty cost into consideration, in CL-RBT, the following multi-objective cost function is used:

$$C(n_i) = \zeta_1 C(n_i \mid n_{root}) + \zeta_2 C_{\text{CostToGo}} + \zeta_3 J(n_i.\Sigma) \quad (19)$$

where C_{CostToGo} denotes the estimated cost-to-go from n_i to the goal state, $C(n_i|n_{root})$ is accumulated along the path from root to n_i, and the first two items correspond to the path-execution cost. $J(n_i.\Sigma)$ is the cost associated with the uncertainty of node n_i, which represents the state uncertainty resulting from following path $\{n_{root} \ldots n_i\}$. ζ_1, ζ_2, ζ_3 are weight factors for adjusting the relative importance of finding a short duration path and reducing state estimation uncertainty.

Since there is a unique path from the root to each of the nodes in the tree, the cost of a node can be used to represent the cost of the associated trajectory. The total cost of the node is used to identify the current best trajectory for execution in the selection and execution procedure, which is presented in the next section.

5.4. Path Selection and Execution

In order to account for the changes in time-varying operating environments, the motion planner operates by periodically selecting and executing the best portion of the current trajectory while continuously expanding the tree during the execution process. The path selection and execution process is presented in Algorithm 3.

Denote the period for the path selection and execution as Δt. At the beginning of each iteration period, the algorithm updates the current actual state of the MAV, as well as the environment information. The purpose of this step is to ensure the MAV's situational awareness (Line 4). After that, the state is first propagated to the end of the period, resulting in $\bar{x}(t_0 + \Delta t)$ (Line 4). Then, the current root of the tree is set to the most current node that is followed by the propagated state, and all other children nodes of the previous root are removed. This is because the MAV is considered to have passed the previous root, and the paths of its children nodes will never be executed. During each of the iterations, the planner identifies the best portion of the trajectory in terms of the cost metric. Since each node n_i in the tree is associated with a single path, the trajectory can be uniquely specified using the sequence consisting of all of the

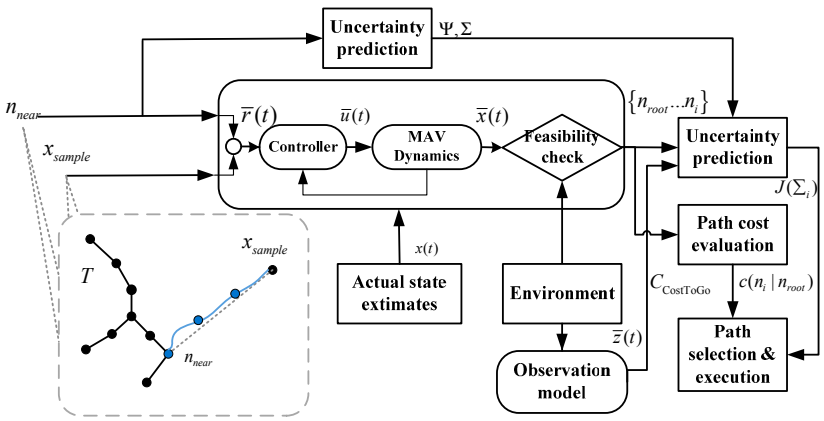
nodes from root to n_i:$\{n_{root}\ldots n_i\}$. In CL-RBT, the cost function given in Equation (19) is used for selecting the current best paths. This multi-objective cost metric enables the motion planner to select paths that achieve a balance between finding low-cost paths to the goal and minimizing localization uncertainty. At the end of each path selection iteration step, the selected feasible trajectory is sent to the controller to be executed, while the tree keeps expanding (Algorithm 1) during the remaining time of period Δt. This mechanism guarantees that the path planning algorithm can provide a feasible path for execution in real time and account for uncertainty in a dynamic environment.

Algorithm 3. Closed-loop random belief tree: execution loop.

1	$t \leftarrow 0$
2	Initialize tree T from node at $x_{initial}$
3	**while** $x(t) \notin X_{goal}$ **do**
4	Update the current actual state $x(t_0)$ and environment information
5	Predict state $x(t)$ to $\bar{x}(t_0 + \Delta t)$ and simulate observations
6	**while** time remaining < Δt **do**
7	expand the tree using Algorithm 1
8	**end while**
9	Select the current best feasible path (node sequence) $p^* = \{n_0 \ldots n_k\}$ according to the multiple-objective cost function given as Equation (19)
10	**if** no best feasible path exists **then**
11	Send brake command to the controller and **goto** 3
12	**else**
13	Send p * to controller to execute
14	**end if**
15	$t \leftarrow t + \Delta t$
16	**end while**

The overall diagram of the CL-RBT algorithm is depicted in Figure 2.

Figure 2. Diagram of the closed-loop random belief tree motion planning framework. MAV, micro-aerial vehicle.



6. Indoor Flight Simulation Results Based on a Simulated Laser Scanner-Equipped MAV

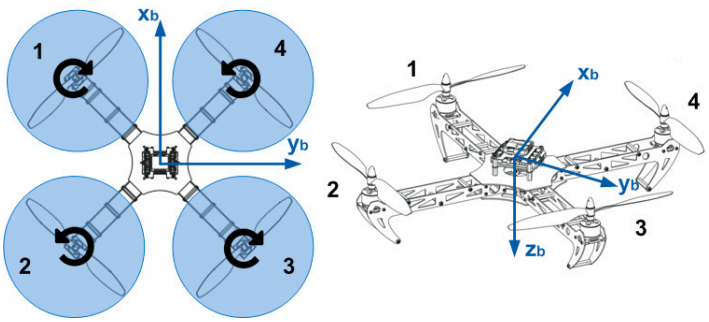
In order to validate the effectiveness of the CL-RBT motion planner, we have conducted a number of experiments in various simulated scenarios that are derived from real-world environments. The first scenario consists of a quadrotor MAV navigating in a 3D wide-open indoor environment, with all structures that can be detected by laser rangefinders concentrated along the peripheral walls. The first scenario is used to demonstrate the CL-RBT's ability to generate the path that ensures the MAV's localization confidence. The subsequent scenarios involve more complex extensions, including 3D unstructured environments (the second scenario), and the purpose of these scenarios is to validate the effectiveness of the CL-RBT algorithm in generating paths that satisfy multiple constraints imposed by dynamic, unstructured environments and non-linear MAV dynamics, while achieving a trade-off between minimizing path cost and reducing localization uncertainty in GPS-denied environments.

For all of these experiments, we assume that the dynamics model of the quadrotor is non-linear and that the environment is without access to GPS signals. The quadrotor must start from an initial spot and traverse through the obstacles to a goal location, relying on a path-following control law and the state estimates provided by an EKF estimator. The exteroceptive sensor equipped on the quadrotor is assumed to be a simulated laser rangefinder with a maximum sensing range of 2 m and a 240° field-of-view. For each of these environments, a 3D model of the environment is first generated using an RGB-D-based environmental modeling approach from our previous work [26]. The 3D point-cloud model is then transformed into a polyhedron-based model, which is used for path planning. The CL-RBT is implemented in MATLAB® and all simulations are performed in real time on an Intel 3.2 GHz platform with a 4 G RAM. The CL-RBT algorithm selects and executes the best parts of the paths every Δt seconds.

6.1. Quadrotor Dynamics Model

The quadrotor model that is used in the simulation is depicted in Figure 3. For the simulation experiments, the x-y-z body-fixed coordinates of the quadrotor is derived using the square configuration, and the kinematic and dynamic model of the quadrotor is developed as rigid-body dynamics influenced by the gravity and thrust of the rotors, assuming near-hover aerodynamics and neglecting the effects caused by the translational velocity. The pure pursuit reference law [27] is applied to steer the reference trajectory, and a PD control law is used to control the quadrotor tracking of the reference. This closed-loop system model consisting of the dynamics model and the control law is used for both the trajectory prediction of the CL-RBT planner and the execution of the resultant trajectory.

Figure 3. Structure and configuration of the quadrotor MAV.



The configuration of the quadrotor utilized in this paper is illustrated in Figure 3. Two coordinate systems are considered in our scheme: the navigation frame (Earth-fixed frame) ($\mathbf{x}_E$, $\mathbf{y}_E$, $\mathbf{z}_E$) and the body-fixed frame ($\mathbf{x}_b$, $\mathbf{y}_b$, $\mathbf{z}_b$). The translational and rotational motions of the quadrotors are achieved through the forces and moments caused by varying the angular rates of the four propellers. Assuming that the quadrotor structure is rigid and symmetrical and the center of mass coincides with the origin of the body-fixed frame, the equations of motion of the quadrotor can be derived using Newton's law and Euler equations [27]. Define φ, θ, ψ as the Euler angles denoting the roll, pitch and yaw attitude, respectively, the rotational dynamics can be described as:

$$\begin{aligned}
\ddot{\varphi} &= \dot{\theta}\dot{\psi}\left(\frac{I_{yy}-I_{zz}}{I_{xx}}\right)+\frac{l}{I_{xx}}b(\omega_3^2+\omega_4^2-\omega_1^2-\omega_2^2)-\frac{K_{drx}}{I_{xx}}\dot{\varphi}+\frac{J_r}{I_{xx}}\dot{\theta}(\omega_2+\omega_4-\omega_1-\omega_3)\\
\ddot{\theta} &= \dot{\varphi}\dot{\psi}\left(\frac{I_{zz}-I_{xx}}{I_{yy}}\right)+\frac{l}{I_{yy}}b(\omega_2^2+\omega_3^2-\omega_1^2-\omega_4^2)-\frac{K_{dry}}{I_{yy}}\dot{\theta}-\frac{J_r}{I_{yy}}\dot{\varphi}(\omega_2+\omega_4-\omega_1-\omega_3)\\
\ddot{\psi} &= \dot{\varphi}\dot{\theta}\left(\frac{I_{xx}-I_{yy}}{I_{zz}}\right)+\frac{1}{I_{zz}}d(\omega_1^2+\omega_3^2-\omega_4^2-\omega_2^2)-\frac{K_{drz}}{I_{zz}}\dot{\psi}
\end{aligned} \tag{20}$$

where I_{xx}, I_{yy}, I_{zz} are the moments of inertial around the three axis, J_r is the rotor inertia and l denotes the length of the moment arm. ω_1, ω_2, ω_3, ω_4 are the four propellers' rotation speeds. b and d denote the rotor thrust coefficient and rotor drag coefficient, respectively. K_{drx}, K_{dry}, K_{drz} represent the rotational drag coefficients of the quadrotor body.

Define (x, y, z) as the three-dimensional position with respect to the navigation frame; the translational dynamics of the quadrotor can be given as:

$$\begin{aligned}
\ddot{x} &= (\cos\varphi\cos\psi\sin\theta+\sin\varphi\sin\psi)\frac{1}{m}\sum_{i=1}^{4}T_i-\frac{K_{dtx}}{m}\dot{x}\\
\ddot{y} &= (\sin\psi\sin\theta\cos\varphi-\sin\varphi\cos\psi)\frac{1}{m}\sum_{i=1}^{4}T_i-\frac{K_{dty}}{m}\dot{y}\\
\ddot{z} &= \cos\theta\cos\varphi\frac{1}{m}\sum_{i=1}^{4}T_i-\frac{K_{dtz}}{m}\dot{z}-g
\end{aligned} \tag{21}$$

where T_i denotes the thrust generated by propeller i and $T_i = b\omega_1^2$, m is the total mass of the quadrotor and g represents the gravitational acceleration. K_{drx}, K_{dry}, K_{drz} are the translational drag coefficients of the quadrotor body.

6.2. Control Scheme Design of the MAV Model

6.2.1. Cascaded Control Scheme

Under the assumption of near-hovering velocities, the drag terms caused by the translational and rotational motions in Equations (20) and (21) can be neglected, and the system dynamics model can be described in the following state-space form ($\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$):

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\theta}\dot{\psi}\left(\frac{I_{yy}-I_{zz}}{I_{xx}}\right)+\frac{l}{I_{xx}}bu_2 \\ \dot{\varphi}\dot{\psi}\left(\frac{I_{zz}-I_{xx}}{I_{yy}}\right)++\frac{l}{I_{yy}}bu_3 \\ \dot{\varphi}\dot{\theta}\left(\frac{I_{xx}-I_{yy}}{I_{zz}}\right)+\frac{1}{I_{zz}}du_4 \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ (\cos\varphi\cos\psi\sin\theta+\sin\varphi\sin\psi)\frac{1}{m}u_1 \\ (\sin\psi\sin\theta\cos\varphi-\sin\varphi\cos\psi)\frac{1}{m}u_1 \\ \cos\theta\cos\varphi\frac{1}{m}u_1-g \end{bmatrix} \tag{22}$$

where $\mathbf{x} = \begin{bmatrix} \varphi & \theta & \psi & \dot{\varphi} & \dot{\theta} & \dot{\psi} & x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$ denotes the state vector, and $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T$ represents the control input:

$$\mathbf{u} = \begin{bmatrix} b(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2) \\ b(\omega_3^2+\omega_4^2-\omega_1^2-\omega_2^2) \\ b(\omega_2^2+\omega_3^2-\omega_1^2-\omega_4^2) \\ d(\omega_1^2+\omega_3^2-\omega_4^2-\omega_2^2) \end{bmatrix} \tag{23}$$

The physical meaning of the control input is related to the forces generated by the four propellers: u_1 represents the total thrust of the propellers, while u_2, u_3 and u_4 are the differences of the propeller pairs.

As can be seen from Equation (20), the rotational dynamics (φ, θ, ψ) are independent of the translational dynamics (x, y, z), leading to decoupled system dynamics. In addition, the altitude dynamics (z) can also be decoupled from the planar translational dynamics (x, y) by assuming that the φ, θ angles are very small. As a result, the quadrotor dynamics in Equation (21) can be decoupled into independent sub-system dynamics, and the cascaded control systems for the attitude, position and altitude can be designed from the decoupled dynamics. Under the assumption of small φ, θ angles and near-hovering motions, we can derive the following linearized attitude dynamics:

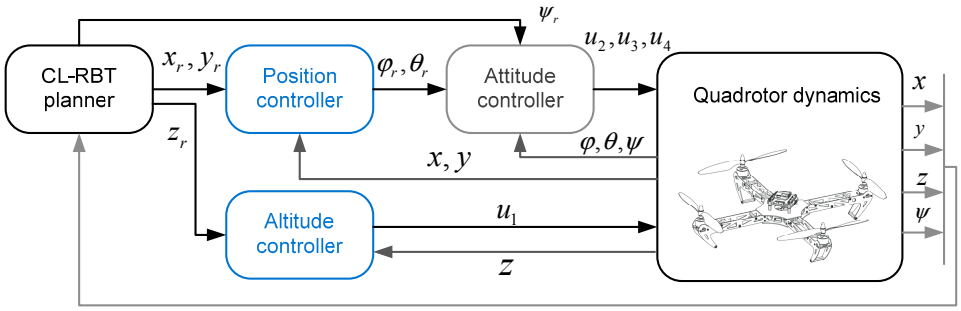
$$\begin{cases} \ddot{\varphi} = \frac{l}{I_x}bu_2 \\ \ddot{\theta} = \frac{l}{I_y}bu_3 \\ \ddot{\psi} = \frac{d}{I_z}u_4 \end{cases} \tag{24}$$

The linearized translational motion dynamics can be given as:

$$\begin{cases} \ddot{x} = (\theta\cos\psi+\varphi\sin\psi)\frac{1}{m}u_1 \\ \ddot{y} = (\theta\sin\psi-\varphi\cos\psi)\frac{1}{m}u_1 \\ \ddot{z} = \frac{1}{m}u_1-g \end{cases} \tag{25}$$

Using the decoupled rotational and translational dynamics in Equations (24) and (25), we can design a cascaded control scheme consisting of the inner control loop and outer control loop, of which the control laws can be designed separately. As illustrated in Figure 4, the outer control loop is the position and altitude controller, which receives the reference path from the CL-RBT planner and generates the reference roll and pitch commands (φ_r, θ_r) to the inner attitude control loop, while the attitude controller produces the desired rotor speed to achieve the reference attitude according to the attitude commands φ_r, θ_r, ψ_r.

Figure 4. Cascaded control scheme of the quadrotor MAV. CL-RBT, closed-loop random belief trees.



6.2.2. Controller Design

Based on the attitude and translational motion dynamics described in Equations (24) and (25), the proportional-integral-derivative (PID) control scheme is utilized to design the attitude and position controller for the quadrotor MAV. The first step is to consider the inner loop controller, which functions as the core of the control scheme. Using the attitude dynamics shown in Equation (24), the attitude controller can be designed as follows:

$$\begin{aligned} u_2 &= K_P^{\varphi,\theta}(\varphi_r - \varphi) + K_I^{\varphi,\theta}\int(\varphi_r - \varphi)dt + K_D^{\varphi,\theta}(\dot{\varphi}_r - \dot{\varphi}) \\ u_3 &= K_P^{\varphi,\theta}(\theta_r - \theta) + K_I^{\varphi,\theta}\int(\theta_r - \theta)dt + K_D^{\varphi,\theta}(\dot{\theta}_r - \dot{\theta}) \\ u_4 &= K_P^{\psi}(\psi_r - \psi) + K_I^{\psi}\int(\psi_r - \psi)dt + K_D^{\psi}(\dot{\psi}_r - \dot{\psi}) \end{aligned} \tag{26}$$

where $K_P^{\varphi,\theta}, K_I^{\varphi,\theta}, K_D^{\varphi,\theta}, K_P^{\psi}, K_I^{\psi}, K_D^{\psi}$ are the control gains of the PID controller and φ_r, θ_r and ψ_r denote the reference roll, pitch and yaw, respectively. Note that φ_r, θ_r are generated by the outer control loop while ψ_r is directly provided by the CL-RBT planner.

Under the assumption of near-hovering velocities and small rotational angle motions, the control scheme of translational dynamics can be decoupled into the planar (x, y) and altitude controllers that can be designed separately. The altitude controller is designed based on PID control with non-linear compensation:

$$u_1 = \frac{1}{\cos\theta\cos\varphi}(K_P^z(z_r - z) + K_I^z\int(z_r - z)dt + K_D^z(\dot{z}_r - \dot{z})) \tag{27}$$

where K_P^z, K_I^z, K_D^z are the PID control gains, and z_r is the reference altitude of the path provided by the CL-RBT planner. The planar position controllers take the following form:

$$\theta_r = \cos\psi[K_P^x(x_r - x) + K_I^x\int(x_r - x)dt + K_D^x(\dot{x}_r - \dot{x})] + \sin\psi[K_P^y(y_r - y) + K_I^y\int(y_r - y)dt + K_D^y(\dot{y}_r - \dot{y})]$$

$$\phi_r = \sin\psi[K_P^x(x_r - x) + K_I^x\int(x_r - x)dt + K_D^x(\dot{x}_r - \dot{x})] - \cos\psi[K_P^y(y_r - y) + K_I^y\int(y_r - y)dt + K_D^y(\dot{y}_r - \dot{y})]$$

where $K_P^x, K_I^x, K_D^x, K_P^y, K_I^y, K_D^y$ are the PID control gains and x_r, y_r are the reference position provided by the CL-RBT planner. As mentioned before, the outer loop controller outputs θ_r, φ_r to the inner loop controller as the reference pith and roll, which are used in Equation (26) for attitude control.

In our applications, the frequency of the inner loop (attitude controller) is 250 Hz, while the outer loop (translational and altitude controller) runs at 30 Hz. The PID gains of the controllers are tuned through extensive numerical simulations in order to achieve a desirable control performance.

In this paper, we only report simulation results based on the PID control law. The reason for selecting the PID scheme is that it can handle complex dynamics model and it is robust to modeling errors. Note that more complex and advanced control laws can also be adopted in the CL-RBT framework. In addition, other types of MAV dynamics models with associated guidance and control laws can be incorporated to form the closed-loop model in CL-RBT.

6.3. EKF Process Model

In the simulation experiments, the states of the MAV are estimated using an EKF-based estimator. The MAV states to be estimated in the EKF filter include: three-axis position in the global frame $\mathbf{p} = [x, y, z]^T$, MAV orientation represented using Euler angles $[\varphi, \theta, \psi]^T$ (roll, pitch and yaw), three-axis velocity in the global frame $\mathbf{v} = [v_x, v_y, v_z]^T$, as well as the gyroscope bias $\mathbf{b}_\omega = [b_{\omega x}, b_{\omega y}, b_{\omega z}]^T$ and accelerometer bias $\mathbf{b}_f = [b_{fx}, b_{fy}, b_{fz}]^T$. The estimated state vector can be denoted specifically as follows:

$$\mathbf{x} = [\mathbf{p}, \mathbf{v}, \varphi, \theta, \psi, \mathbf{b}_\omega, \mathbf{b}_f]^T = [x, y, z, v_x, v_y, v_z, \varphi, \theta, \psi, b_{\omega x}, b_{\omega y}, b_{\omega z}, b_{fx}, b_{fy}, b_{fz}]^T \quad (28)$$

As described previously, the MAV is equipped with a laser rangefinder, which is used by scan-matching to provide position and heading measurements (x, y, ψ). The altitude of the MAV is measured by an onboard sonar altimeter, and it is assumed that the altitude measurement is not affected by the attitude of the MAV. In addition to the laser rangefinder, the MAV is assumed to be equipped with an IMU module, which consists of a triaxial gyroscope and triaxial accelerometer. The triaxial gyroscope provides three-axis angular rates $\omega_m = [\omega_{mx}, \omega_{my}, \omega_{mz}]^T$, while the accelerometer measures the three-axis accelerations $f_m = [f_{mx}, f_{my}, f_{mz}]^T$, and the above IMU measurements are all expressed in the MAV's body frame. Using the above definitions, the continuous state-space-based process model ($\mathbf{x} = g(\mathbf{x}, \mathbf{u})$) describing the IMU dynamics can be given by:

$$\dot{\mathbf{p}} = \mathbf{v} \quad (29)$$

$$\dot{\mathbf{v}} = \tilde{C}_b^n(f_m - \mathbf{b}_f) - \mathbf{g} \quad (30)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} \omega_{mx} - b_{\omega x} \\ \omega_{my} - b_{\omega y} \\ \omega_{mz} - b_{\omega z} \end{bmatrix} \quad (31)$$

$$\dot{\mathbf{b}}_{\omega} = 0 \tag{32}$$

$$\dot{\mathbf{b}}_{f} = 0 \tag{33}$$

where $\tilde{C}_b^n$ denotes the transformation matrix from the body frame to the global frame, and $\mathbf{g} = [0, 0, g]^{\mathrm{T}}$ is the local gravity vector in the global frame. Since the x, y position and the heading ψ (yaw) of the MAV are estimated separately using the measurements from the laser scan-matching, ψ can be treated independent of roll φ and pitch θ, and the MAV motion of the x-, y-axes can be decoupled from z-axis motion under the assumption of near-hovering velocity and small φ, θ angles. This assumption yields the simplified transformation matrix in Equation (30):

$$\tilde{C}_b^n = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{34}$$

A similar approach for the simplification of the MAV motion can also be found in [28,29] (Note that although this simplified velocity model may cause significantly large estimation error when the MAV performs aggressive attitude maneuvers or high-speed flight, it is sufficiently accurate for state estimation under the above assumptions in this paper).

To implement the state estimation on a computer system, the above process model is transformed into a discrete-time model. Let Δt be the update period of EKF and assume that Δt is sufficiently small, the process model in Equations (29)–(33) can be discretized as follows:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \Delta t f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \tag{35}$$

In order to implement the EKF estimator, the above discrete process model needs to be further linearized by calculating the Jacobians $\partial g / \partial \mathbf{x}|_{\mathbf{x}_{t-1}}$ as described in Equation (2). To derive the Jacobians, the calculation of partial derivatives of the process model with respect to the state vector is given as follows:

$$G = \left.\frac{\partial g}{\partial \mathbf{x}}\right|_{\mathbf{x}_{t-1}} = I + \Delta t \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}_{t-1}} = \begin{bmatrix} I_{3\times3} & \Delta t I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & \Delta t G_{23} & 0_{3\times3} & \Delta t G_{25} \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} + \Delta t \mathrm{G}_{33} & \Delta t G_{34} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} \end{bmatrix} \tag{36}$$

where:

$$G_{23} = \begin{bmatrix} 0 & 0 & -\sin\psi(f_{mx} - b_{fx}) - \cos\psi(f_{my} - b_{fy}) \\ 0 & 0 & \cos\psi(f_{mx} - b_{fx}) - \sin\psi(f_{my} - b_{fy}) \\ 0 & 0 & 0 \end{bmatrix}, \; G_{25} = \begin{bmatrix} -\cos\psi & \sin\psi & 0 \\ -\sin\psi & -\cos\psi & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$G_{33} = \begin{bmatrix} (\omega_{my} - b_{\omega y})\cos\varphi\tan\theta - (\omega_{mz} - b_{\omega z})\sin\varphi\tan\theta & (\omega_{my} - b_{\omega y})\sin\varphi\sec^2\theta + (\omega_{mz} - b_{\omega z})\cos\varphi\sec^2\theta & 0 \\ -(\omega_{my} - b_{\omega y})\sin\varphi - (\omega_{mz} - b_{\omega z})\cos\varphi & 0 & 0 \\ (\omega_{my} - b_{\omega y})\cos\varphi\sec\theta - (\omega_{mz} - b_{\omega z})\sin\varphi\sec\theta & (\omega_{my} - b_{\omega y})\sin\varphi\sin\theta\sec^2\theta + (\omega_{mz} - b_{\omega z})\cos\varphi\sin\theta\sec^2\theta & 0 \end{bmatrix}$$

$$G_{34} = \begin{bmatrix} -1 & -\sin\varphi\tan\theta & -\cos\varphi\tan\theta \\ 0 & -\cos\varphi & \sin\varphi \\ 0 & -\sin\varphi\sec\theta & -\cos\varphi\sec\theta \end{bmatrix}$$

The process noise w is introduced by the IMU angular rates (ω_m) and accelerations (f_ω), which are included as the input $\mathbf{u}$ to the process model:

$$\mathbf{u} = \left[\omega_m, f_m,\right]^{\mathbf{T}}, w = \left[w_\omega, w_f\right]^{\mathrm{T}} \tag{37}$$

where w_ω, w_f are the noise associated with the angular rates and accelerations, respectively, and w_ω, w_f are both white Gaussian noises with zero mean, *i.e.*, $w_\omega \sim N(0, Q_\omega)$, $w_f \sim N(0, Q_f)$. Similarly, the partial derivatives of the process model with respect to the process noise can be calculated by:

$$V_t = \left.\frac{\partial g}{\partial w}\right|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} = \Delta t \left.\frac{\partial f}{\partial w}\right|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & \Delta t V_{31} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & \Delta t V_{22} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix}^{\mathrm{T}} \tag{38}$$

where:

$$V_{22} = \tilde{C}_b^n, V_{31} = \begin{bmatrix} 1 & \sin\varphi\tan\theta & \cos\varphi\tan\theta \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi\sec\theta & \cos\varphi\sec\theta \end{bmatrix}$$

Due to the different measurement rates of the IMU, the sonar and the scan-matching module, the state and covariance are updated separately at different times, using three groups of measurements. The pitch, roll angle (φ, θ) and the gyroscope bias $\mathbf{b}_\omega$ are updated first using the measurements from the IMU (more specifically, the accelerometer measurements). The altitude z is updated separately using the sonar measurement, since it is uncorrelated with the other measurements. In addition, the update of the MAV position $\mathbf{p}$, heading ψ, the velocity v_x, v_y and the accelerometer bias $\mathbf{b}_f$ are completed when the measurements from laser scan-matching are received. By the above discussions, while the performance of pitch and roll (φ, θ) estimates depends on the IMU, the estimation of x, y position and the heading ψ are primarily affected by the characteristics of the laser rangefinder, as well as the perceived environment information. As a result, in order to evaluate how the path planning strategy affects the uncertainty of state estimation, we extract the laser-related states $\tilde{\mathbf{x}} = [x, y, v_x, v_y, \psi, b_{fx}, b_{fy}]^{\mathbf{T}}$ from the full state vector. By Equation (35), the process update model of $\tilde{\mathbf{x}}$ can be given as:

$$\begin{aligned}
x_t &= x_{t-1} + v_{x|t-1}\Delta t \\
y_t &= y_{t-1} + v_{y|t-1}\Delta t \\
\begin{bmatrix} v_x \\ v_y \end{bmatrix}_t &= \begin{bmatrix} v_x \\ v_y \end{bmatrix}_{t-1} + \Delta t \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}_{t-1} \begin{bmatrix} f_{mx} - b_{fx} \\ f_{my} - b_{fy} \end{bmatrix}_{t-1} + \Delta t \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}_{t-1} \begin{bmatrix} w_{fx} \\ w_{fy} \end{bmatrix} \\
\psi_t &= \psi_{t-1} + \Delta t(\omega_{mz|t-1} - b_{\omega z|t-1}) + \Delta t w_{\omega z} \\
b_{fx|t} &= b_{fx|t-1} \\
b_{fy|t} &= b_{fy|t-1}
\end{aligned} \tag{39}$$

By the above process model, we have:

$$\tilde{\mathbf{x}}_t = \tilde{g}(\tilde{\mathbf{x}}_{t-1}, \tilde{\mathbf{u}}_{t-1}, \tilde{w}) \tag{40}$$

where $\tilde{\mathbf{u}} = \left[f_{mx}, f_{my}, \omega_{mz}\right]^{\mathrm{T}}$, $\tilde{w} = \left[w_{fx}, w_{fy}, w_{\omega z}\right]^{\mathrm{T}}$, and the Jacobians of the above model can be obtained directly from Equations (36) and (38):

$$\tilde{G}_t = \left.\frac{\partial g}{\partial \mathbf{x}}\right|_{\mathbf{x}_{t-1}\mathbf{u}_{t-1}} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -\Delta t \sin\psi(f_{mx}-b_{fx}) - \Delta t \cos\psi(f_{my}-b_{fy}) & -\Delta t\cos\psi & \Delta t \sin\psi \\ 0 & 0 & 0 & 1 & \Delta t \cos\psi(f_{mx}-b_{fx}) - \Delta t \sin\psi(f_{my}-b_{fy}) & -\Delta t\sin\psi & -\Delta t\cos\psi \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (41)$$

$$\tilde{V}_t = \begin{bmatrix} 0 & 0 & \Delta t\cos\psi & \Delta t\sin\psi & 0 & 0 & 0 \\ 0 & 0 & -\Delta t\sin\psi & \Delta t\cos\psi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \Delta t & 0 & 0 \end{bmatrix}^{\mathrm{T}} \quad (42)$$

Denote σ_{fx}^2, σ_{fy}^2 and $\sigma_{\omega z}^2$ as the noise covariance of the accelerometer measurements in the *x*- and *y*-axis, as well as the gyroscope measurements in the *z*-axis, respectively. The process noise covariance of the process model in Equation (39) can be given by:

$$Q = diag(\sigma_{fx}^2, \sigma_{fy}^2, \sigma_{\omega z}^2) \quad (43)$$

Therefore, the matrix S_t in Equation (11) can be calculated by $S_t = \tilde{V}_t Q \tilde{V}_t^T$, and $\tilde{G}_t$ (Equation (41)), $\tilde{V}_t$ (Equation (42)), S_t are used in the linear covariance propagation, as described in Equation (11).

6.4. Sensor Measurement Model and Uncertainty Analysis of Laser Rangefinders

In many actual applications, a MAV performs localization using an onboard laser rangefinder, which obtains measurements through a series of range readings recorded at consecutive angles increments. Therefore, the accuracy of the localization is mainly affected by the range to the objects and the topology of the environments. In our simulation, we assume that the quadrotor is equipped with a laser rangefinder and that the localization is performed by matching the laser scans. In order to incorporate the uncertainty of the laser rangefinder for the covariance propagation in the CL-RBT planner, we used an information matrix-based method [6] to determine the uncertainty of the laser measurements with respect to the topology of the environments.

For a quadrotor using a laser rangefinder, the information matrix *N* in Equation (11) denotes the information provided by laser scans, which can be calculated by combining the information of each range measurement of the environment. Considering an environment geometry shown in Figure 5, a laser scan obtained at time *t* consists of measurements of *n* scan points in the environment, each of which is described by a range reading r_i and a measurement angle θ_i, and the information provided in a laser scan (r_i, θ_i) is in the direction perpendicular to the topology, *i.e.*, the direction of the vector normal (the vector in green, Figure 5) of the environment surface. Therefore, the information matrix can be obtained by projecting the range information onto the vector normal of the surface and summing the projected information of each scan point. The information matrix *N* can be calculated by:

$$\mathbf{N} \triangleq \mathbf{H}^T (\Sigma_r^2)^{-1} \mathbf{H} = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} & \Sigma_{x\psi} \\ \Sigma_{xy} & \Sigma_{yy} & \Sigma_{y\psi} \\ \Sigma_{x\psi} & \Sigma_{y\psi} & \Sigma_{\psi\psi} \end{bmatrix} \quad (44)$$

with **H** as the measurement matrix (Equation (45)) and Σ_r^2 as the variance matrix (Equation (46)) denoting the variance of each measurement:

$$\mathbf{H} = \begin{bmatrix} \cos\gamma_1 \cos(\gamma_1 - \theta_1) & \sin\gamma_1 \cos(\gamma_1 - \theta_1) & r_1 \sin(\gamma_1 - \theta_1) \\ \vdots & \ddots & \vdots \\ \cos\gamma_n \cos(\gamma_n - \theta_n) & \cos\gamma_n \cos(\gamma_n - \theta_n) & r_i \sin(\gamma_n - \theta_n) \end{bmatrix} \tag{45}$$

$$\Sigma_r^2 = diag(\sigma_{r_1}^2, \sigma_{r_1}^2, \sigma_{r_1}^2, ...\sigma_{r_i}^2, \sigma_{ri}^2, \sigma_{r_i}^2 ...\sigma_{r_n}^2, \sigma_{r_n}^2, \sigma_{r_n}^2) \tag{46}$$

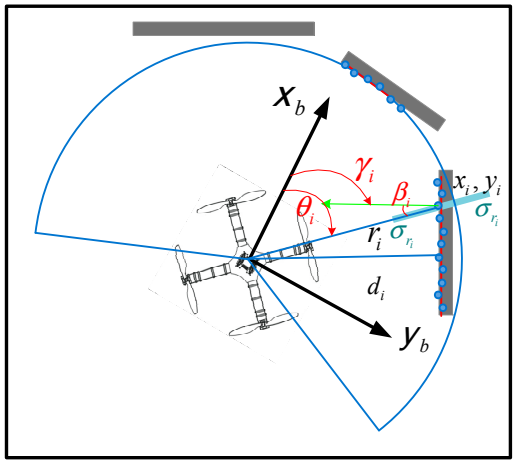
In our experiments, the readings of each measurement point (r_i, θ_i) are obtained first from the simulated laser scan, and then, the line segments of the topology surface are extracted using the measurement points, along with the vector normal of each line segments, as well as the angle between the body axis and the vector normal. From Equations (44)–(46), the information matrix N can be calculated with the summation terms that are given as follows:

$$\Sigma_{xx} = \sum_i^m \frac{\cos^2\gamma_i \cos^2(\gamma_i - \theta_i)}{\sigma_{r_i}^2}, \Sigma_{yy} = \sum_i^m \frac{\sin^2\gamma_i \cos^2(\gamma_i - \theta_i)}{\sigma_{r_i}^2}$$

$$\Sigma_{\psi\psi} = \sum_i^m \frac{r_i^2 \sin^2(\gamma_i - \theta_i)}{\sigma_{r_i}^2}, \Sigma_{xy} = \sum_i^m \frac{\cos\gamma_i \sin\gamma_i \cos^2(\gamma_i - \theta_i)}{\sigma_{r_i}^2}$$

$$\Sigma_{x\psi} = \sum_i^m \frac{r_i \cos\gamma_i \sin(\gamma_i - \theta_i)\cos(\gamma_i - \theta_i)}{\sigma_{r_i}^2}, \Sigma_{y\psi} = \sum_i^m \frac{r_i \sin\gamma_i \sin(\gamma_i - \theta_i)\cos(\gamma_i - \theta_i)}{\sigma_{r_i}^2}$$

Figure 5. Uncertainty analysis of a laser range measurement. The blue points denote the measurement points in a single laser scan at time *t*. γ is the angle between the body axis and the vector normal of the surface at measurement point (x_i, y_i), and β_i is the angle form the vector normal to the measurement direction. d is the distance from the origin to the line segment i.



The above analysis provides a statistical approach to incorporating the variance of the laser scan data into the uncertainty of the pose estimate. Therefore, we can use Equations (44)–(46) to calculate the information matrix N, which is used in the linear covariance propagation as described in Equation (11).

Figure 6a–e show the uncertainty ellipses related to the position estimation uncertainties along a predefined path calculated using the information matrix equations proposed in this section, as well as

the covariance propagation approaches in Section 4. In the experiments, an 18 × 9 m 3D environment model is derived from a cluttered real-world laboratory, and the MAV is assumed to be equipped with a simulated laser rangefinder that is able to generate range and bearing measurements of the environment. In order to highlight the influences of the environment, the laser sensor model is assumed to have a limited sensing capability with a 2-m range and a 240° field-of-view, which is represented by blue sectors in the figures (Although this sensor model is unrealistic in terms of maximum range, it serves well to illustrate how the environment and sensor measurement affect the localization uncertainty). As the MAV navigates along the path and rotates its orientations, the environment topologies that fall within the field-of-view will generate a series of noisy measurements of the topologies' distance and bearing to the MAV body (the valid measurements are demonstrated by the cyan sectors in the following figures), enabling the MAV to localize itself by a EKF-based approach using these measurements of the environment. The 1–σ position uncertainty is demonstrated by red ellipses in the figures, and the size and shape of the ellipses indicate the relative scale of the localization uncertainty in both *x* and *y* directions.

Figure 6. Position estimation uncertainties based on a simulated laser sensor along a predefined path. Cyan sectors denote the laser data with valid environmental measurements. Red ellipses illustrate the 1–*σ* localization covariance; larger ellipses indicate high uncertainty poses. All covariance are in cm. (**a**) position: (5.0, 2.0, 0.5) m, heading: −45°; (**b**) position: (8.0, 7.5, 0.5) m, heading: −45°; (**c**) position: (8.0, 7.5, 0.5) m, heading: 0°; (**d**) position: (11.0, 4.0, 1.0) m, heading: −30°; (**e**) position: (10.0, 2.5, 1.0) m, heading: 0°.

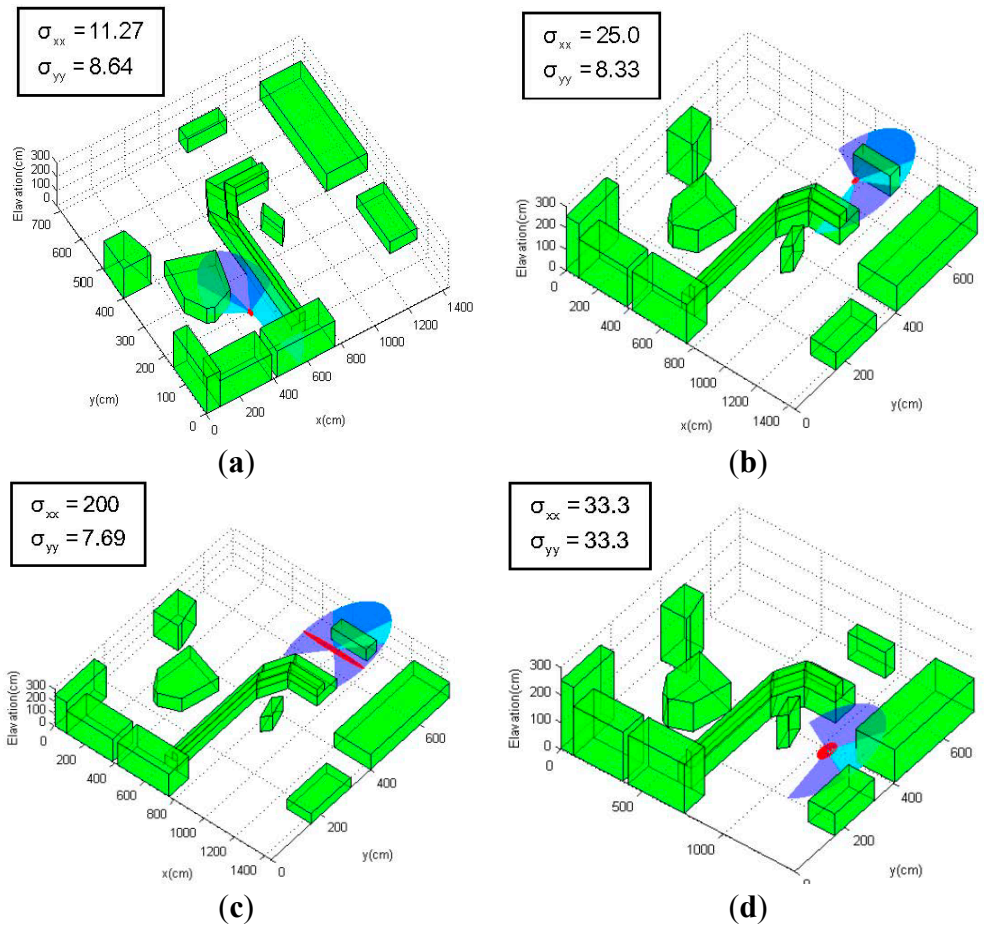
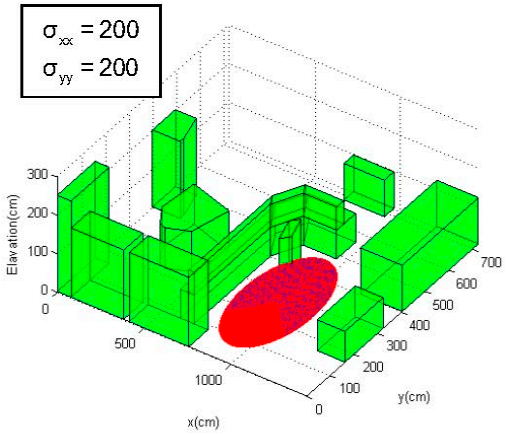


Figure 6. *Cont.*



(**e**)

As can be seen from Figure 6, the area where the onboard sensor is expected to detect more environmental topologies tends to produce position estimates with low uncertainty (Figure 6a,b), while those locations where the sensor only encounters just one obstacle will lead to high localization uncertainty (Figure 6d). In extreme cases where the MAV navigates into wide open areas (Figure 6e), the localization uncertainty increases sharply, since these areas provide almost no information for localization. In addition, the localization uncertainty is also affected by the orientation (yaw) of the MAV relative to the environmental topologies: The MAV is at the same *x*-*y*-*z* coordinates in Figure 6b,c, but the position uncertainty varies due to the different yaw orientations of the sensor (−45° in Figure 6b and 0° in Figure 6c); the yaw angle is positive when the rotation is counterclockwise around the *z*-axis of the Earth-fixed frame). The environmental topology of the MAV location in Figure 6b,c can be considered as a corridor between two parallel obstacles, and the laser sensor measurements will provide most information (large information matrix *N*) in the direction perpendicular to the local environment, according to the model given in Equations (44)–(46). As a result, the uncertainty in the direction of the "corridor" (*x* direction of the environment frame) is much larger when the measurement direction is perpendicular to the "corridor" (Figure 6c).

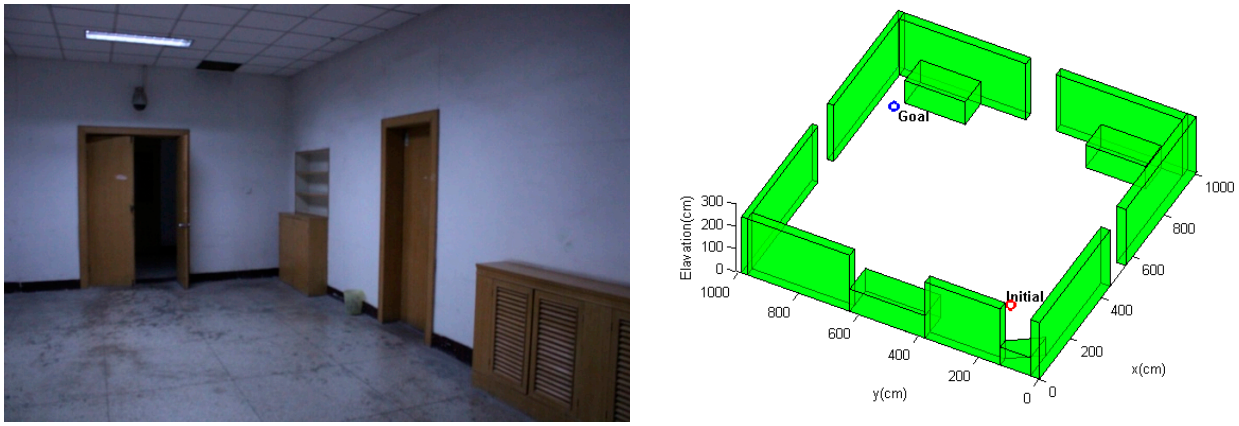
6.5. Numerical Simulation Results

6.5.1. Scenario 1: Indoor Environment with Open Space

The environment of the first scenario is shown in Figure 7a. As can be seen from Figure 7, this scenario is a 10 × 10-m open indoor environment with all structures concentrated along the peripheral walls, and the center of the environment is an open region that is out of the measurement range of the simulated laser rangefinder. The MAV quadrotor must navigate from the initial position at the bottom-right corner through the hallway to a goal region, which is diagonally opposite of the initial point. Since most of the environment provides rare measurement for localization, it is even more critical for the path planning strategy to find paths that maximize information gain and reduce state uncertainty; hence, the emphasis of this simulation scenario is to test the path planning algorithm's ability to generate paths that ensure the MAV's localization performance throughout the path. For comparison purposes,

the conventional CL-RRT algorithm without the incorporation of state uncertainty is also tested using the same MAV model and environment configurations.

Figure 7. Indoor environment simulation Scenario 1: (**a**) A hallway of Tsinghua University's main building; (**b**) simulated 3D model of the GPS-denied indoor environment. The red and blue circles indicate the initial and goal location of the MAV, respectively.



(**a**) (**b**)

In this scenario, the quadrotor MAV begins at $\mathbf{x}_{initial} = [2.0 \quad 2.0 \quad 0.0]^T$ m. It is intended to navigate towards the goal location at $\mathbf{x}_{goal} = [9.0 \quad 9.0 \quad 1.0]^T$ m, and the weight factors in Equation (19) are set as: $\zeta_1 = 1$, $\zeta_1 = 1$, $\zeta_3 = 100$. The cascaded PID control scheme described in Section 6.2 is applied for both CL-RBT state propagation and reference path following of the MAV. The planning-execution cycle is set to 5 s.

Figure 8 demonstrates an example trajectory generated by a trial of the simulation scenario. The CL-RBT algorithm quickly generates a feasible path that goes through the regions with as much measurements as possible to localize and reaches the goal (Figure 8a,b). Since a comparatively small amount of samples and nodes are generated during the initial planning, the initially selected paths may still reach the open regions with comparatively high localization uncertainty (Figure 8c–e). However, as more samples and nodes are added, the CL-RBT continuously refines the path in real time, identifying optimal paths that go through measurement-rich regions to ensure localization and reduce the execution cost of the goal (Figure 8f–h).

In contrast, by expanding the path using the conventional CL-RRT and checking obstacle collision constraints, the resulting paths will move the MAV straight to the goal (Figure 9). Although the conventional CL-RRT may generate shorter paths, ignoring the localization factors will result in high localization uncertainty, since these paths may go through the open regions. In this case, following these paths would likely cause operation failure, since the state estimate becomes unacceptably uncertain, such that the MAV control would have become unstable.

Figure 8. Example results of trajectory generated by CL-RBT for a quadrotor MAV navigating in simulation Scenario 1. The pink ellipses represent the covariance of the position estimate, and large ellipses denote MAV states with high uncertainty. The current state (MAV's 3D positions and headings) is denoted with a red pentacle. The current selected path (specified by node sequence, used as the reference to the control system) is marked in black with red dots representing the nodes. The predicted closed-loop trajectory corresponding to the current selected node sequence is emphasized in red, while the actual output trajectory flown by the closed-loop MAV model is marked in pink. The current tree is denoted by green nodes and cyan trajectories, but is set as invisible after (**b**) for clarity. (**a**) t = 5 s; (**b**) t = 10 s; (**c**) t = 20 s; (**d**) t = 25 s; (**e**) t = 30 s; (**f**) t = 40 s; (**g**) t = 50 s; (**h**) t = 60 s.

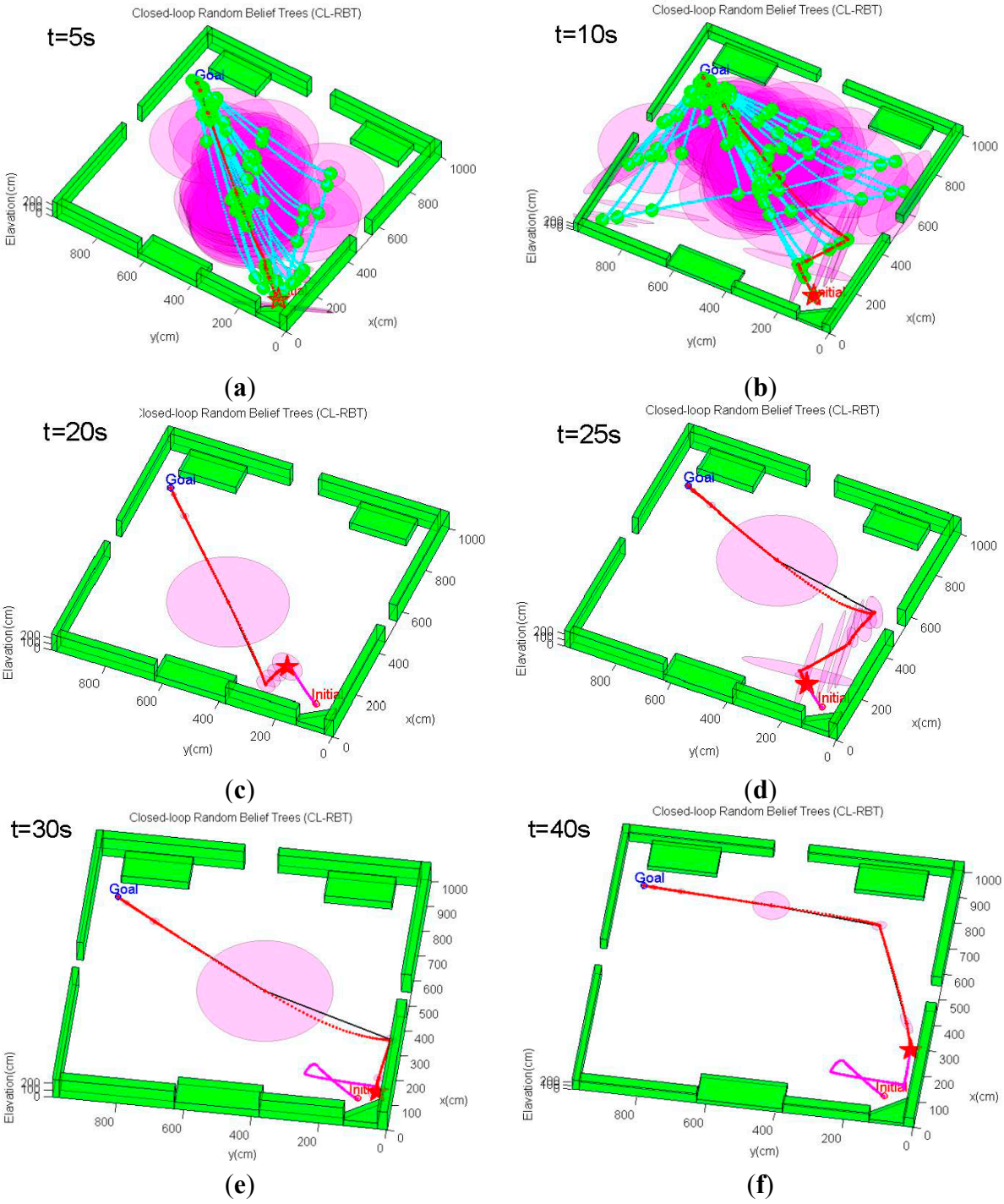
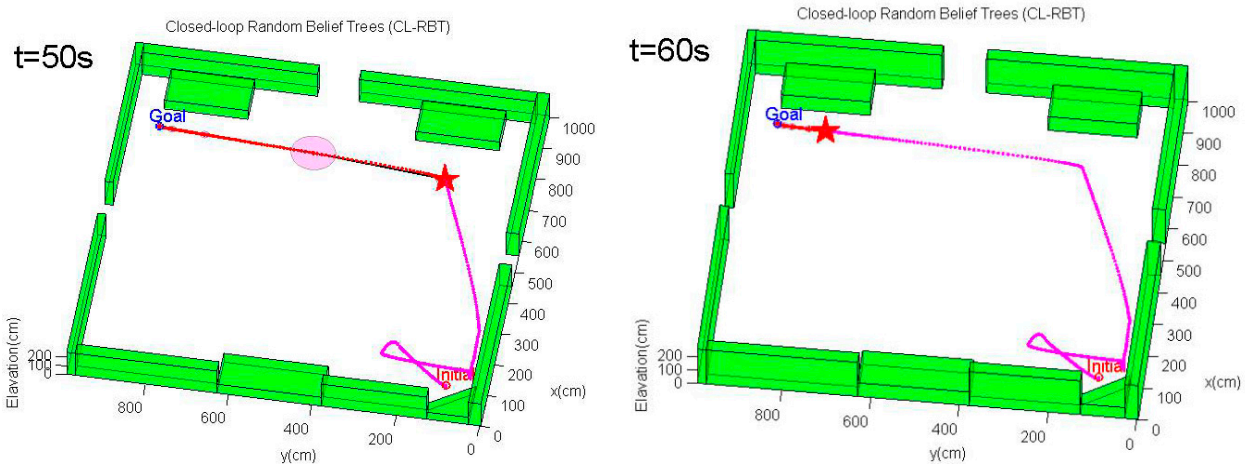
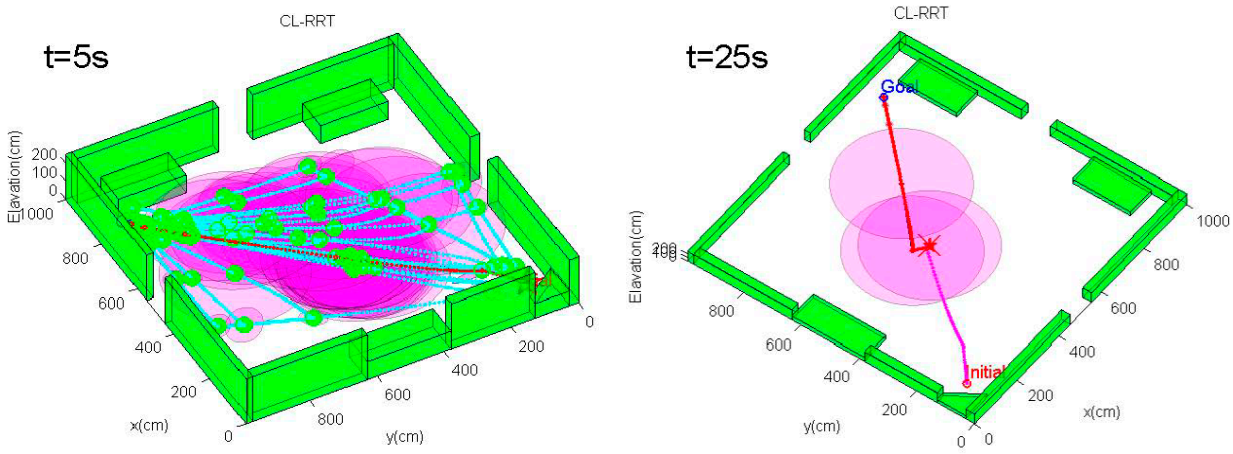


Figure 8. *Cont.*



(g) **(h)**

Figure 9. Example results of the trajectory generated by the conventional closed-loop rapidly exploring random trees (CL-RRT) for a quadrotor MAV navigating in simulation Scenario 1. The red cross in (**b**) denotes the state where the MAV's state estimation fails, since the position uncertainty has become sufficiently high. (**a**) t = 5 s; (**b**) t = 25 s.

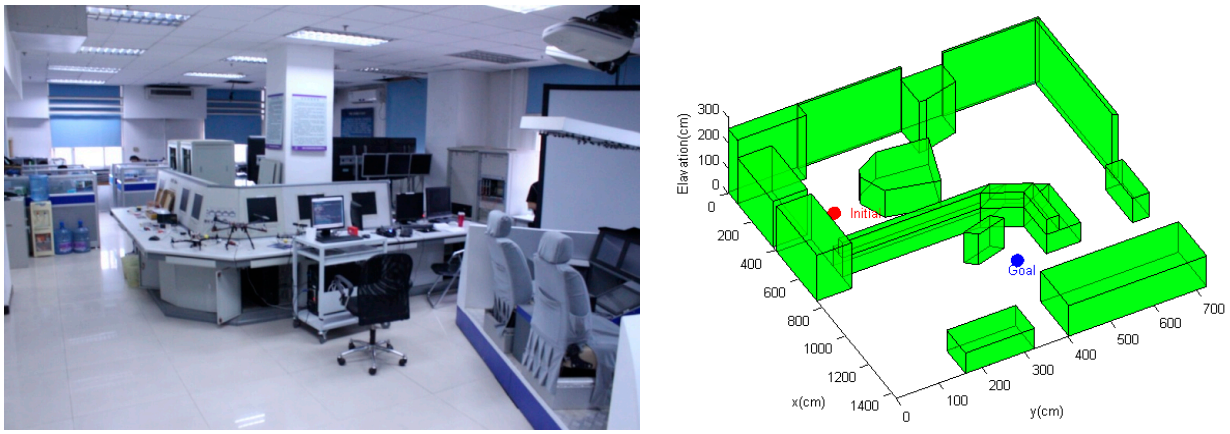


(a) **(b)**

6.5.2. Scenario 2: Cluttered 3D Indoor Environment

This scenario considers a more complex three-dimensional, GPS-denied indoor environment with a number of non-convex obstacles and structures (Figure 10). The environment model is derived from an actual cluttered laboratory, which is the same as the model shown in Figure 6. The objective of the quadrotor MAV is to navigate from the bottom-left initial point to the goal position located on the other side of the laboratory, moving safely between the obstacles. Planning trajectories in this cluttered environment is a challenging task for the path planner. The unstructured environment requires the path planning algorithm to be able to generate a trajectory that avoids the non-convex obstacles to ensure feasibility, while bounding the state estimate uncertainty along the trajectory to allow the MAV to remain well-localized.

Figure 10. Indoor environment simulation Scenario 2: (**a**) view of a typical cluttered laboratory; (**b**) simulated 3D model of the unstructured, GPS-denied indoor environment. The red and blue dot indicate the initial and goal location of the MAV, respectively.



(**a**) (**b**)

For this scenario, the quadrotor MAV begins from one side of the environment at $\mathbf{x}_{initial} = [3.2 \quad 6.4 \quad 0.0]^T m$, and it is intended to navigate to the goal point behind a cluttered region of obstacles at $\mathbf{x}_{goal} = [10.0 \quad 4.0 \quad 0.5]^T m$. The same MAV dynamics model, PID control scheme and sensor configuration as in Scenario 1 are also applied in this scenario. However, we select weighting factors in Equation (19) as $\zeta_1 = 1$, $\zeta_1 = 20$, $\zeta_3 = 30$, since the emphasis of this scenario is on generating feasible trajectories that quickly reach the goal while reducing localization uncertainty. The planning cycle in this scenario is extended to 7 s. The conventional CL-RRT is also tested using the same scenario and MAV system model for comparison purpose.

An example of the trajectory simulation results from this scenario is depicted in Figure 11. As can be seen from Figure 11, the MAV's initial position is located in an obstacle-rich region. The path planner initially has to move the MAV out of this cluttered region. Unlike the first scenario, the CL-RBT algorithm does not find any trajectories directly to the goal due to the poor coverage of the CL-RBT tree and the cluttered environment in the first few planning cycles (Figure 11a,b). As the CL-RBT tree expands to cover more space after some wandering, the CL-RBT algorithm moves the MAV progressively through the narrow passages and approaches towards the direction of the goal region Figure 11c). After moving out of the narrow passage region, the path planner succeeds in identifying a trajectory to the goal (Figure 11d). As can be seen from the figure, the covariance of the position estimate stays bounded along the trajectory during the entire planning cycles of CL-RBT.

In contrast to CL-RBT, the conventional CL-RRT quickly finds a path that goes over the cluttered obstacle region and directly reaches the goal (Figure 12a). Although the resulting path is shorter than the path generated by CL-RBT, the uncertainty of the position estimate grows rapidly along the CL-RRT trajectory (Figure 12b); this is because the environment becomes open as the MAV's altitude increases, providing rare information to the onboard sensor for localization.

Figure 11. Example results of the trajectory generated by the CL-RBT for a quadrotor MAV navigating in a cluttered environment. The full legend for the symbols can be found in Figure 8. The current tree is set as invisible after (**b**) for clarity. (**a**) t = 7 s; (**b**) t = 14 s; (**c**) t = 21 s; (**d**) t = 28 s; (**e**) t = 35 s; (**f**) t = 42 s; (**g**) t = 56 s; (**h**) t = 70 s.

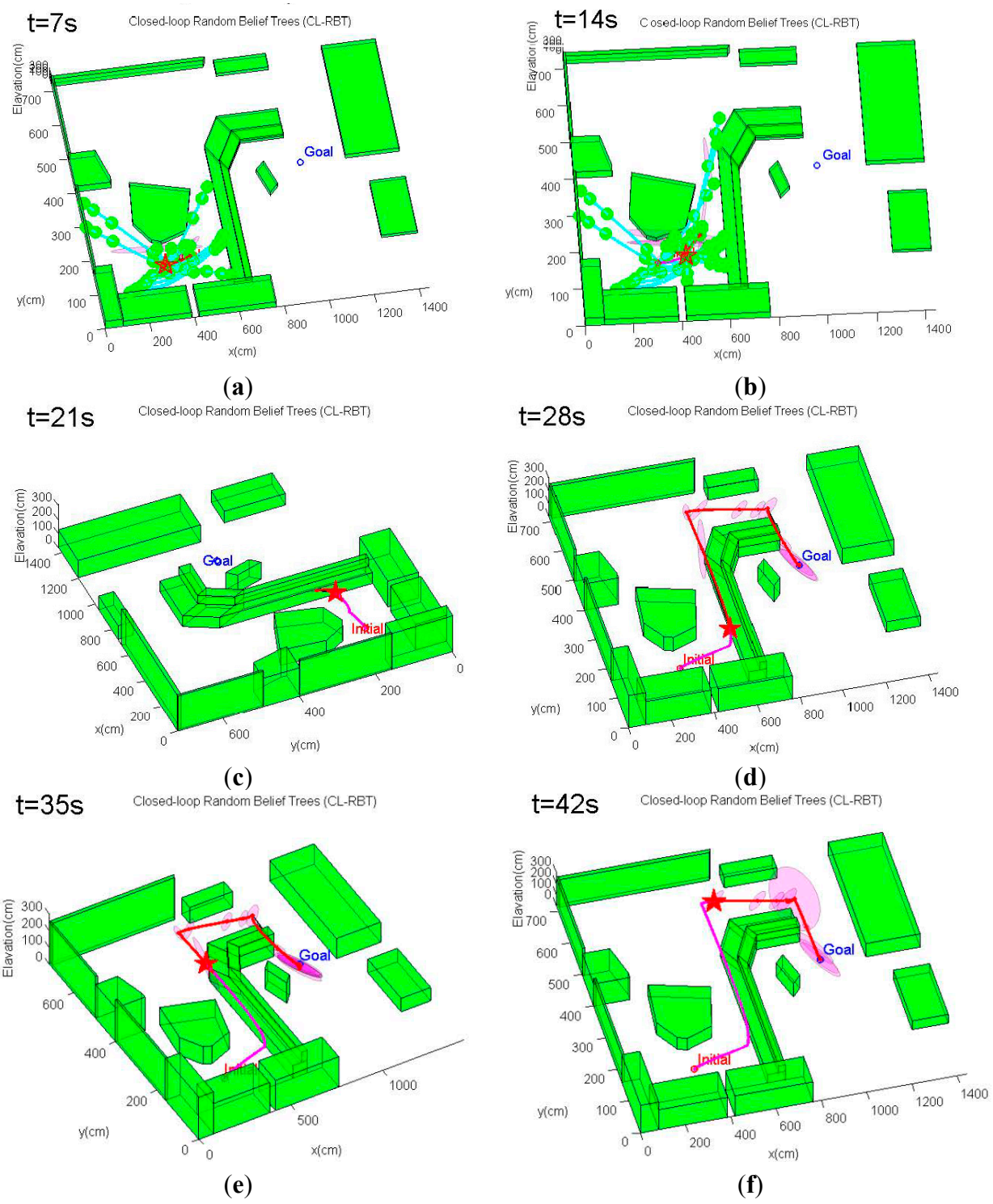
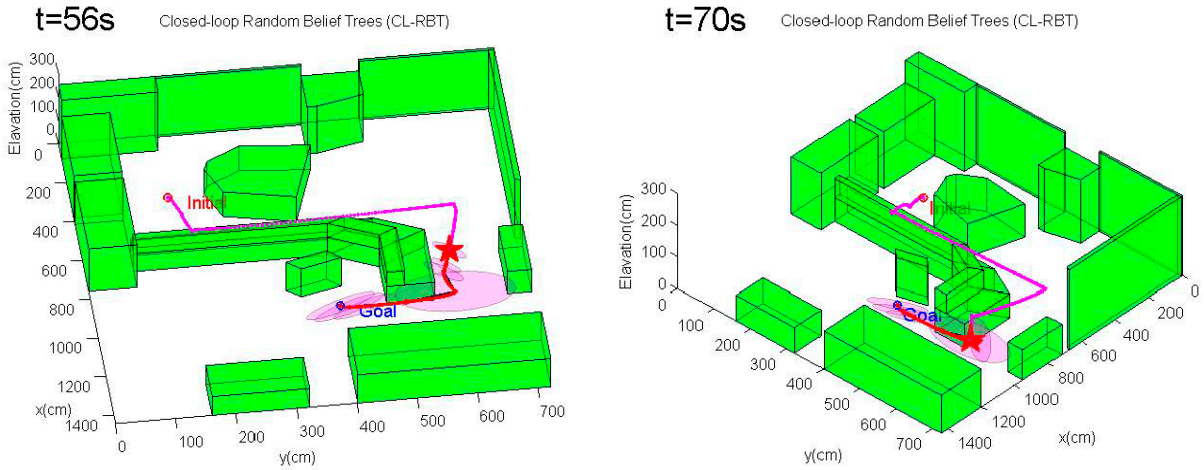
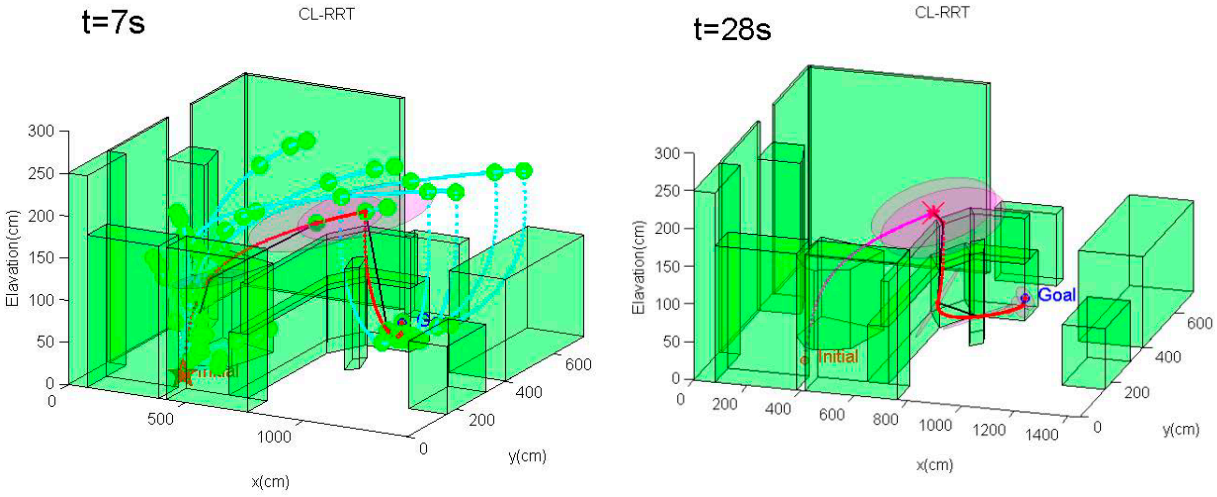


Figure 11. *Cont.*



(**g**) (**h**)

Figure 12. Example results of trajectory generated by the conventional CL-RRT for a quadrotor MAV navigating in simulation Scenario 1. The red cross in (**b**) denotes the position where the MAV's state estimation fails since the position uncertainty have exceeded the threshold. (**a**) t = 7 s; (**b**) t = 28 s.



(**a**) (**b**)

6.5.3. Performance Comparison and Analysis

In order to illustrate the performance of the CL-RBT algorithm, we compare the performance statistics of the CL-RBT algorithm to that of the conventional CL-RRT by running simulation experiments using both algorithms on the same scenarios, as described in Sections 6.5.1 and 6.5.2 (note that we did not compare the CL-RBT to other conventional sampling-based path planners that operate on static graphs or trees, such as PRM, RRT or BRM, since they are not real-time algorithms in essence). For each simulation scenario, the experiment is repeated 30 times using each of the two algorithms, and the performance statistics is recorded and evaluated in terms of the paths' cost (total length) and the trace of the MAV's expected covariance at the goal when following the generated paths, both averaged over 30 times.

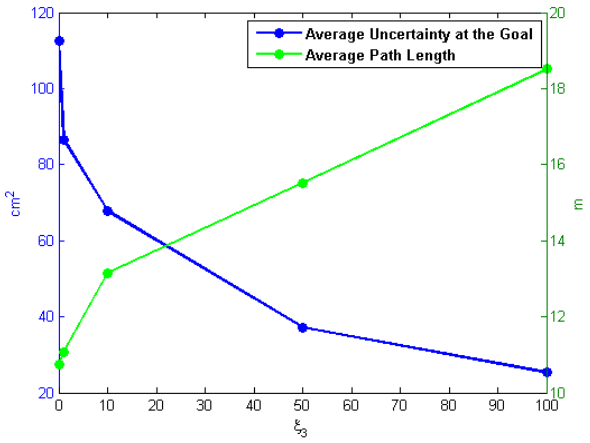
Table 1 shows a comparison of the average path length and covariance statistics of CL-RBT and conventional CL-RRT. The CL-RBT outperforms conventional CL-RRT in terms of expected uncertainty at the goal in both simulation scenarios. Moreover, since these experiments are performed in simulation, the actual trajectories of the MAV are available, and the average mean error between the estimated MAV position and the actual position at the goal location are also computed and shown in Table 1 (note that the mean errors corresponding to the conventional CL-RRT are computed by assuming that the controller is able to execute the prescribed path and navigate the MAV to the goal location regardless of the estimation error, while in fact, the MAV controller would have failed halfway). These results indicate that the conventional CL-RRT results in large deviations from the true position, while the CL-RBT can effectively bound the estimation error in repeated trials. However, the resulting paths generated by CL-RRT are generally longer that those of conventional CL-RRT. This is because CL-RBT integrates the prediction of position uncertainty into the planning progress, which enabling the MAV to balance minimizing path length against reducing localization uncertainty, selecting longer paths to avoid regions that lead to high position estimation uncertainty when necessary.

Table 1. Performance comparisons of CL-RBT and CL-RRT, using 30 simulation runs.

Algorithms	Parameters	Scenario 1	Scenario 2
CL-RBT	Path length (m)	17.35	16.7
	Final covariance (cm^2)	**39.88**	**41.65**
	Final mean error (m)	1.07	0.82
CL-RRT	Path length (m)	10.74	12.78
	Final covariance (cm^2)	5,933	9,523
	Final mean error (m)	54.33	73.52

As discussed previously, the specification of weighting factors in the cost function for path selection Equation (19) is an important factor for implementations of CL-RBT. To evaluate the effect of the selection on weighting factors, we have conducted a number of simulations with different values of weighting factor ζ_3 in Scenario 1. Five different values of ζ_3 are evaluated in these simulations: $\zeta_3 = \{0, 1, 10, 50, 100\}$. For each of the five values, the simulation is repeated 10 times, and the path length and uncertainty cost (trace of the covariance) at the goal are recorded and averaged. Figure 13 illustrates the resulting paths' average length and uncertainty cost at the goal as a function of ζ_3. As can be seen from the figure, the uncertainty cost decreases as the weighting factor ζ_3 increases, whereas the path becomes comparatively longer. This is because reducing localization uncertainty becomes relatively more significant as ζ_3 increases, resulting in longer paths that take more measurements from the environment. These results demonstrate that the weighting factors of the cost function should be carefully selected depending on the specific task requirement and the characteristic of the environment.

Figure 13. Comparison of average path length and average uncertainty cost at the goal. Data points correspond to different values of the uncertainty cost weighting factor (Equation (19)): $\zeta_3 = \{0, 1, 10, 50, 100\}$.



7. Conclusions and Future Work

This paper presents a real-time path planning approach in belief space (CL-RBT) for MAVs with complex dynamic constraints under state estimation uncertainty. The proposed CL-RBT approach is built upon the RRT motion planning framework. We made two primary contributions in this paper. First, the prediction of state uncertainty is incorporated in the path planning process, which allows the path planning strategy to find the path that reduces the state uncertainty and ensures state estimation confidence, using an efficient, linear update process of covariance in a factored form. Second, closed-loop prediction is used in the motion planning framework to generate dynamically feasible trajectories, enabling the motion planner to handle complex dynamic and environment constraints. Simulation results demonstrate that the motion planner can be implemented in real time, generating dynamically feasible paths that trade off minimizing path cost and reducing state uncertainty accuracy, while easily handling complex vehicle dynamics. It can also be concluded that CL-RBT has the potential to increase the autonomy of MAVs that operate in complex, GPS-denied environments.

While the utility and advances of the CL-RBT algorithm have been presented in this paper, there still remains significant future work on the proposed framework. Future work will focus on the more extensive analysis of the algorithm's performance, including the computational complexity and optimality of the generated paths. Further theoretical work is also necessary in the proof and analysis of the algorithm's completeness and convergence. Moreover, we are also planning to implement the CL-RBT on an actual quadrotor MAV platform that is currently under development. It would be possible to validate the performance of CL-RBT with realistic onboard sensors (laser rangefinders, RGB-D, cameras, *etc.*) through flight tests in actual indoor environments.

Acknowledgments

This work is sponsored by the China High-Tech 863 Program, No. 2001AA415340 and No. 2007AA04Z1A6, the China Natural Science Foundation, No. 61174168, the Aviation Science Foundation of China, No. 20100758002 and 20128058006.

Author Contributions

D.L. proposed the CL-RBT approach, conducted the simulation experiments and wrote the paper. Q.L. and J.S. supervised this work, and N.C. provided advices on the derivation of the linear covariance propagation approach and the establishment of the EKF process model.

Conflicts of Interest

The authors declare no conflicts of interest.

References

1. Bachrach, A.; He, R.; Roy, N. Autonomous Flight in Unknown Indoor Environments. *Int. J. Micro Air Veh.* **2009**, *4*, 277–298.
2. Shaojie S.; Michael, N.; Kumar, V. Autonomous Indoor 3D Exploration with a Micro-Aerial Vehicle. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; IEEE Press: New York, NY, USA, 2012.
3. Bry, A.; Bachrach, A.; Roy, N. State Estimation for Aggressive Flight in GPS-denied Environments Using Onboard Sensing. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; IEEE Press: New York, NY, USA, 2012.
4. Abdelkrim, N.; Aouf, N.; Tsourdos, A.; White, B. Robust Nonlinear Filtering for INS/GPS UAV Localization. In Proceedings of the Mediterranean Conference on Control and Automation, Ajaccio, France, 25–27 June 2008; IEEE Press: New York, NY, USA, 2008.
5. Michael, N.; Fink, J.; Kumar, V. Cooperative Manipulation and Transportation with Aerial Robots. *Auton. Robot.* **2010**, *30*, 73–86.
6. Chowdhary, G.; Sobers, D.M.; Pravitra, C.; Christmann, C.; Wu, A.; Hashimoto, H.; Ong, C.; Kalghatgi, R.; Johnson, E.N. Self-Contained Autonomous Indoor Flight with Ranging Sensor Navigation. *J. Guid. Control Dyn.* **2012**, *35*, 1843–1854.
7. Weiss, S.; Scaramuzza, D.; Siegwart, R. Monocular-SLAM based Navigation for Autonomous Micro Helicopters in GPS-denied Environments. *J. Field Robot.* **2011**, *28*, 854–874.
8. Huang, A.S.; Bachrach, A; Henry, P.; Krainin, M.; Maturana, D.; Fox, D.; Roy, N. Visual Odometry and Mapping for Autonomous Flight Using an Rgb-d Camera. In Proceedings of the International Symposium on Robotic Research, Flagstaff, AZ, USA, 28 August–1 September 2011; pp. 1–16.
9. Frazzoli, E.; Dahleh, M.A.; Feron, E. Real-time motion planning for agile autonomous vehicles. *J. Guid. Control Dyn.* **2002**, *25*, 116–129.
10. Samuel, P.; Roy, N. The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance. *Int. J. Robot. Res.* **2009**, *28*, 1448–1465.
11. Thrun, S.; Burgard, W.; Fox, D. Chap. 15. Partially Observable Markov Decision Process. In *Probabilistic Robotics*; MIT Press: Cambridge, MA, USA 2005; pp. 513–542.
12. Kaelbling, L.; Littman, M.; Cassandra, A. Planning and Acting in Partially Observable Stochastic Domains. *Artif. Intell.* **1998**, *101*, 99–134.
13. Berg, V.D.; Patil, J.S.; Alterovitz, R. Motion Planning under Uncertainty Using Iterative Local Optimization in Belief Space. *Int. J. Robot. Res.* **2012**, *31*, 1263–1278.

14. Bai, H.; Hsu, D.; Lee, W.; Ngo, V. Monte Carlo Value Iteration for Continuous State POMDPs. In Proceedings of the 9th International Workshop on the Algorithmic Foundations of Robotics, Singapore, Singapore, 13–15 December 2010; Springer: Berlin/Heidelberg, Germany, 2011.
15. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580.
16. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Technical Report; Iowa State University: Ames, IA, USA, 1998.
17. Kuwata, Y.; Teo, J.; Karaman, S.; Fiore, G.; Frazzoli, E.; How, J.P. Motion Planning in Complex Environments Using Closed-loop Prediction. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Honolulu, HI, USA, 18–21 August 2008.
18. Kuwata, Y.; Teo, J.; Fiore, G.; Karaman, S.; Frazzoli, E.; How, J.P. Real-time motion planning with applications to autonomous urban driving. *IEEE Trans. Control Syst. Technol.* **2009**, *17*, 1105–1118.
19. Berg, V.D.; Abbeel, P.; Goldberg, K. LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information. *Int. J. Robot. Res.* **2011**, *30*, 895–913.
20. Bry, A.; Roy, N. Rapidly-exploring Random Belief Trees for Motion Planning under Uncertainty. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; IEEE Press: New York, NY, USA, 2011.
21. He, R.; Prentice, S.; Roy, N. Planning in Information Space for a Quadrotor Helicopter in a Gps-denied Environment. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; IEEE Press: New York, NY, USA, 2008.
22. Abraham, B.; Prentice, S.; He, R.; Henry, P.; Huang, A.S.; Krainin, M.; Maturana, D.; Fox D.; Roy, N. Estimation, Planning, and Mapping for Autonomous Flight Using an RGB-D Camera in GPS-denied Environments. *Int. J. Robot. Res.* **2012**, *31*, 1320–1343.
23. Levine, D.; Luders, B.; How, J.P. Information-rich path planning with general constraints using rapidly-exploring random trees. In Proceedings of the AIAA Infotech@ Aerospace Conference, Atlanta, GA, USA, 20–22 April 2012.
24. Kalman, E.R. A New Approach to Linear Filtering and Prediction Problems. *J. Fluid. Eng.* **1960**, *82*, 35–45.
25. Smith, R.; Self, M.; Cheeseman, P. Estimating Uncertain Spatial Relationships in Robotics. In *Autonomous Robotic Vehicles*; Cox, I.J., Wilfong, G.T., Eds.; Springer: Orlando, FL, USA, 1990; pp. 167–193.
26. Qing, L.; Dachuan, L.; Qifan, W.; Liangwen, T.; Yan, H.; Yixuan, Z.; Nong, C. Autonomous navigation and environment modeling for MAVs in 3-D enclosed industrial environments. *Comput. Ind.* **2013**, *64*, 1161–1177.
27. Park, S.; Deyst, J.; How, J.P. Performance and Lyapunov Stability of a Nonlinear Path-Following Guidance Method. *J. Guid. Control Dyn.* **2007**, *30*, 1718–1728.
28. Bachrach, A.; Prentice, S.; He, R.; Roy, N. RANGE—Robust autonomous navigation in GPS-denied environments. *J. Field Robot.* **2011**, *28*, 644–666.
29. Bachrach, A. Autonomous Flight in Unstructured and Unknown Indoor Environments. Master's Thesis, Massachusetts Institute of Technology: Cambridge, MA, USA, September 2009.