2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Stochastic Automatic Collision Avoidance for Tele-Operated Unmanned Aerial Vehicles

Daman Bareiss Jur van den Berg Kam K. Leang

***Abstract*—This paper presents a stochastic approach for automatic collision avoidance for tele-operated unmanned aerial vehicles (UAVs). Collision detection and mitigation in the presence of uncertainty is an important problem to address because on-board sensing and state estimation uncertainties are inherent in real-world systems. A feedforward-based algorithm is described that continually extrapolates the future trajectory of the vehicle given the current operator control input for collision avoidance. If the predicted probability of a collision is greater than a user-defined confidence bound, the algorithm overrides the operator control input with the nearest, safe command signal to steer the robot away from obstacles, while maintaining user intent. The algorithm is implemented on a simulated quadrotor helicopter (quadcopter) with varying amounts of artificial uncertainty. Simulation results show that for a given confidence bound, the aerial robot is able to avoid collisions, even in a situation where the operator is deliberately attempting to crash the vehicle.**

I. Introduction

The number of civil and commercial applications for unmanned aerial vehicles (UAVs) has risen tremendously over the past few decades. The applications include environmental control and monitoring [1], 3D mapping [2], telecommunication [3], crop and aquaculture farm monitoring [4], unexploded ordnance detection [5], traffic monitoring [6], and media resources [7]. Since many small to medium sized multi-rotor UAVs have the ability to access hard to reach indoor and outdoor locations or areas that are unfit for humans, they can be used for search and rescue, law enforcement, or first responders to enhance situational awareness [8], [9]. However, one of the most daunting tasks for even a skilled UAV pilot is collision avoidance, especially in tight and compact environments such as inside of a partially collapsed building where usually the only feedback is a live-camera feed. Thus, automatic collision avoidance technology for tele-operated UAVs is critical and necessary to allow pilots to focus on higher-priority tasks such as locating survivors.

In this paper, a feedforward-based collision avoidance algorithm that considers sensing and estimation uncertainties while maintaining the user's intent is presented [see block diagram in Fig. 1(b)]. Specifically, a collision is avoided by exploiting the dynamics of the robot and the measured relative distances between objects in the environment for automatically determining control inputs to safely steer the UAV away from obstacles [10]. To enable the use of UAVs in real-world applications, UAVs will need to be equipped with on-board sensors to measure and/or estimate the distance to nearby obstacles and maintain an internal estimate of their state [11]. However, when on-board sensing technology, such as light detection and ranging (LIDAR), is used the uncertainty in the sensor's output can significantly affect the performance of the collision avoidance algorithm. For example, the popular Hokuyo RG-04LX-UG01 LIDAR range sensor most commonly used in robotics for obstacle avoidance has an accuracy up to $\pm 3\%$ of the measurement. Measurement error combined with uncertainty in state estimation (due to the fact that a model of the robot's dynamics are used) can lead to collisions. Because sensing and state estimation uncertainties are inherent in real-world applications, it is necessary to consider these uncertainties when developing collision avoidance algorithms.

The contribution of this work is a feedforward-based collision avoidance algorithm that explicitly considers uncertainty in the location of the obstacles and uncertainty in the robot

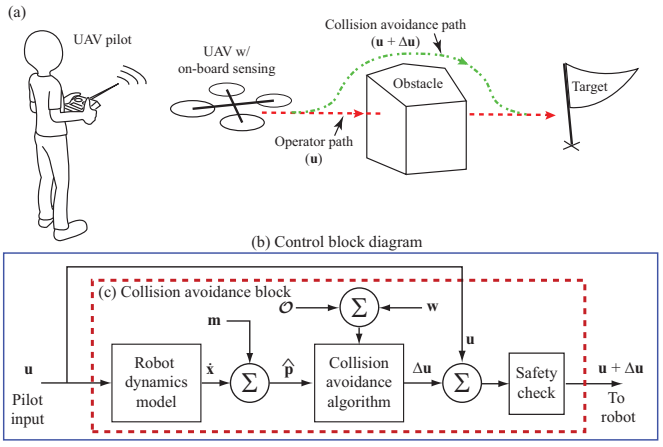

Fig. 1. Collision avoidance for tele-operated UAVs: (a) concept where UAV pilot controls the aircraft and provides a user input $\mathbf{u}$. When a collision is detected, with on-board sensing and state estimation, the algorithm produces an input $\Delta\mathbf{u}$ that augments the pilot's input $\mathbf{u}$ to steer the robot away from the obstacle. (b) Control block diagram and (c) the collision avoidance block where the pilot's input $\mathbf{u}$ is passed through the dynamics model to obtain the estimated trajectory $\hat{\mathbf{p}}$ along with uncertainty in the motion model $\mathbf{m}$. This trajectory is checked for collisions against the obstacles $\mathcal{O}$ including uncertainty $\mathbf{w}$. If a collision is detected, the algorithm calculates a change in input $\Delta\mathbf{u}$ to avoid collisions. If the input $\mathbf{u} + \Delta\mathbf{u}$ is deemed safe, it is then passed to the robot.

D. Bareiss and K. K. Leang are with the Design, Automation, Robotics & Control (DARC) Lab, Department of Mechanical Engineering at the University of Utah. E-mails: daman.bareiss@utah.edu and kam.k.leang@utah.edu.

J. van den Berg is with the School of Computing at the University of Utah. E-mail: berg@cs.utah.edu.

model [see collision avoidance block in Fig. 1(c)]. In particular, the proposed method estimates the trajectory of the robot from the current time into the future for some predetermined amount of time given the robot's dynamics and the control input (from the operator). This trajectory is checked for any collisions with the obstacles in the environment given the uncertainty in the estimated trajectory and the uncertainty in the obstacle location, where an on-board LIDAR sensor could be used for obstacle detection. If the probability of a collision is found to be above some confidence bound, the algorithm determines a new control input that is as close as possible to the operator's original input while avoiding collisions through a convex optimization. The minimal change in the input allows the method to maintain the user's intent as much as possible while avoiding collisions.

The feedforward-based algorithm is implemented on a simulated quadrotor helicopter in a variety of environments with different magnitudes of artificial uncertainty added to the obstacle detection and trajectory estimation. It is demonstrated in simulation that the algorithm provides collision-free motion probabilistically given the confidence bound selected.

The remainder of this paper is structured as follows. Related work and comparisons of the proposed work with similar techniques are discussed in Sec. II. The problem is defined in Sec. III, followed by a detailed presentation of the stochastic collision avoidance algorithm in Sec. IV. Simulation details, results, and discussion are presented in Sec. V. Finally, concluding remarks and future work are presented in Sec. VI.

II. Related Work

Collision avoidance is an important research topic in robotics, where numerous approaches have been developed and applied to manufacturing systems [12], medical devices [13], and mobile service robots [14]. Some early methods include potential fields [15], [16], the dynamic window approach [17], velocity obstacles [18], and vector field histograms (VFH) [14].

In general, collision avoidance methods can be classified into one of two main categories: global and local (reactive methods). First, collisions between a mobile robot and obstacles can be achieved through a motion planning algorithm [19]–[23] which typically assumes *a priori* information about the environment. These methods search the robot's possible trajectories for the best trajectory with respect to some goal, typically choosing a trajectory that minimize the uncertainty. These methods share the similarity that they select a trajectory and define the control inputs to control the robot optimally along the selected trajectory. Often, global planners are computationally expensive and information about the environment is required.

A second class of collision avoidance algorithms are local or reactive methods. These methods do not optimize a trajectory, but rather they find a change in control input that will approximately avoid collisions given a local knowledge (sensor information) of the obstacles. In many of the reactive algorithms uncertainty is often handled by improving sensory perception [24], [25] or using relative sensing information and developing control laws that guarantee separation between agents (and obstacles) in the presence of uncertainty [26]. Additionally, algorithms also approximate the noise by artificially increasing the size of the robot empirically based on the uncertainty [27]. Other techniques deal with state uncertainty by exploiting dynamic programming [28].

Integrated global and local planners have been explored, where proposed algorithms use a predicted trajectory to avoid collisions with the observable, local obstacle [18]. Typically, these algorithms avoid collisions by computing a given change in input for a current sensing-action cycle, but limited work has explicitly considered uncertainty and those that do typically add a buffer or safety zone around the robot [24]–[27] . The approach in this paper also exploits both global and local information, but considers explicitly the uncertainty in the estimation and measurement process. By using a feedforward prediction of the flight path and computing the expected robot position along the trajectory, the proposed method can perform a less approximate consideration of noise than increasing the radius of the robot arbitrarily as in reactive planners, but less exact than a full trajectory optimization of the global planners. This results in an approximate, but reliable and robust method that can operate in real-time to assist UAV pilots with collision avoidance.

III. Problem Formulation

A. Notation

In the following, vector sets are denoted using calligraphics, for example $\mathcal{A}$. Vectors are represented by boldface fonts, such as $\mathbf{a}$; matrices are denoted by upper case italics, for example A; and scalars are represented by lower-case italics, such as a. Scalar and matrix multiplications, and Minkowski sums of sets are defined as $a\mathcal{B} = \{a\mathbf{b} \mid \mathbf{b} \in \mathcal{B}\}$, $A\mathcal{B} = \{A\mathbf{b} \mid \mathbf{b} \in \mathcal{B}\}$, $\mathcal{A} \oplus \mathcal{B} = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{B}\}$.

A vector $\mathbf{a}$ sampled from a multivariate normal distribution with mean $\boldsymbol{\mu}$ and variance Σ, where Σ is positive-semidefinite, is denoted by $\mathbf{a} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$.

B. System equations, uncertainty, workspace, and obstacles

Consider a robot with general, nonlinear equations of motion and a state space of arbitrary dimension m. Let $\mathcal{X} \subset \mathbb{R}^m$ be the state space of the robot and let $\mathcal{U} \subset \mathbb{R}^n$ be the control input space of the robot. Let the continuous-time equations of motion of the robot be defined by the function $\mathbf{f} \in \mathcal{X} \times \mathcal{U} \to \mathbb{R}^m$,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{m}, \qquad \mathbf{m} \sim \mathcal{N}(\mathbf{0}, M), \quad (1)$$

where $\mathbf{x}(t) \in \mathcal{X}$ and $\mathbf{u}(t) \in \mathcal{U}$ are the state and control input at time t, respectively. It is assumed that the motion of the robot is corrupted by zero-mean Gaussian noise $\mathbf{m} \in \mathbb{R}^m$ with a given covariance $M \in \mathbb{R}^{m \times m}$, where M is positive semi-definite.

For a given input $\mathbf{u}$, the predicted state $\hat{\mathbf{x}}(t)$ follows

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}, \mathbf{u}, t). \tag{2}$$

Given an initial state $\mathbf{x} = \mathbf{x}(0)$, $\hat{\mathbf{x}} = \hat{\mathbf{x}}(0)$, and a constant input $\mathbf{u}$, the state of the robot for $t > 0$ is defined by

$$\mathbf{x}(t) \sim \mathcal{N}(\hat{\mathbf{x}}(t), P(t)), \tag{3}$$

where $\hat{\mathbf{x}}(t) = \mathbf{g}(\hat{\mathbf{x}}, \mathbf{u}, t)$ is the expected state at time t, $\mathbf{g} \in \mathcal{X} \times \mathcal{U} \times \mathbb{R} \to \mathcal{X}$ represents the solution to $\mathbf{f}$ in Eq. (1)). $P(t)$ is the uncertainty of the state, defined as

$$P(t) = \mathrm{E}\left[(\mathbf{x}(t) - \hat{\mathbf{x}}(t))(\mathbf{x}(t) - \hat{\mathbf{x}}(t))^T\right]. \tag{4}$$

The uncertainty at time t, Eq. (4), is found by solving the following differential equation:

$$\dot{P}(t) = A(t)P(t) + P(t)A(t)^T + M, \tag{5}$$

where

$$A(t) = \frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}}(\hat{\mathbf{x}}(t), \mathbf{u}(t)). \tag{6}$$

Let $\mathbb{R}^d$ be the workspace in which the robot maneuvers, where typically $d \leq 3$, and let $\mathcal{O} \subset \mathbb{R}^d$ define the subset of the workspace occupied by obstacles. In order to maintain compatibility with the implementation of on-board sensing, those regions of the workspace that are occluded by the obstacles as seen from the current state of the robot are also considered obstacles. In other words, the subspace of the workspace that cannot be seen by the robot is also an obstacle. The obstacles' positions can be defined relative to the robot's position including uncertainty with known variance $Z \in \mathbb{R}^{d \times d}$

$$\mathcal{O} = \bigcup_{i=1}^{n} \mathcal{O}_i \oplus \{\mathbf{w}\}, \quad \mathbf{w} \sim \mathcal{N}(0, Z), \tag{7}$$

where $\mathcal{O}_i$ represents individual objects in the environment that are considered obstacles and $\mathbf{w}$ is drawn from a zero-mean normal distribution with variance Z. The relative obstacle paradigm provides the benefit of eliminating the need for the robot to maintain an estimate of its position over time.

Let $\mathcal{R}(\mathbf{x}) \subset \mathbb{R}^d$ denote the subset of the workspace occupied by the robot when it is in state $\mathbf{x} \in \mathcal{X}$. Then, a colliding state is defined as

$$\mathcal{R}(\mathbf{x}(t)) \cap \mathcal{O} \neq \emptyset. \tag{8}$$

C. Problem Definition

The problem is defined as finding a minimal change $\Delta\mathbf{u} \in \mathcal{U}$ to the control input $\mathbf{u} \in \mathcal{U}$ given the initial state $\mathbf{x} \in \mathcal{X}$ of the robot to avoid collisions with obstacles within a time horizon τ. The probability that the robot will collide with an obstacle given the variance in the state estimate and obstacle location must be less than a predetermined value $\bar{p}$ for all time less than the time horizon $\tau \in \mathbb{R}$, hence

$$\text{minimize: } \Delta\mathbf{u}^T R \Delta\mathbf{u} \tag{9}$$

$$\text{subject to: } p\left(\forall t \in [0, \tau] :: \mathcal{R}(\mathbf{x}(t)) \cap \mathcal{O} = \emptyset \mid P(t), Z\right) \leq \bar{p},$$

where $R \in \mathbb{R}^{n \times n}$ is a positive-definite weight matrix, $P(t)$ is the variance in the forward prediction of the state, and Z is the variance in the obstacle location.

IV. A Stochastic Approach to Collision Avoidance

A. Approach

In the following, a feedforward approach is presented for collision avoidance [see block diagram in Fig. 1(b)]. First, the following assumptions are made to simply the nonlinear, non-convex optimization problem for real-time implementation.

Assumption 1. *The robot's position $\mathbf{p} \in \mathbb{R}^d$ can be derived from the state through a projection*

$$\mathbf{p}(t) = C\mathbf{x}(t), \tag{10}$$

where $C \in \mathbb{R}^{d \times m}$.

Assumption 2. *The geometry of the robot $\mathcal{R}$ is defined as the smallest enclosing sphere centered at its reference point such that the geometry is rotationally invariant. Let $\mathcal{R}(\mathbf{p}) \subset \mathbb{R}^d$ be the spherical subset of the workspace occupied by the robot at position $\mathbf{p}$.*

Assumption 3. *The robot's trajectory can be represented through a first-order Taylor expansion, i.e.,*

$$\hat{\mathbf{p}}(t, \Delta\mathbf{u}) \approx \hat{\mathbf{p}}^\star(t) + J(t)\Delta\mathbf{u}, \tag{11}$$

where

$$\hat{\mathbf{p}}^\star(t) = C\mathbf{g}(\hat{\mathbf{x}}, \mathbf{u}, t), \; J(t) = C\frac{\partial \mathbf{g}}{\partial \mathbf{u}}(\hat{\mathbf{x}}, \mathbf{u}, t). \tag{12}$$

Assumption 4. *If the robot is collision-free at time τ with respect to an appropriately chosen convex subset of the free workspace, then it is assumed that the robot is also collision-free for all time $t \in [0, \tau]$. This is reasonable for relatively short time horizons τ.*

Given the robot's current state $\mathbf{x}$ and the current control input $\mathbf{u}$ (from the operator), the estimated positions of the robot in the future are found by Eq. (11). The variance on the predicted state was given in Eq. (4).

From Assumption 1, the mapping from the robot's state to its position also defines the variance on the robot's position, i.e.,

$$P_c(t) = CP(t)C^T. \tag{13}$$

From Assumption 4, when there is uncertainty in both the obstacle location and the robot's estimated trajectory, a probability for a collision must be considered rather than a deterministic collision or collision-free state. Thus, given independent Gaussian distributions representing the uncertainty in the trajectory estimation and the obstacle location, respectively, the probability for a collision is non-zero if

$$\mathcal{N}(\hat{\mathbf{p}}^\star(\tau), P_c(\tau) + Z) \cap \mathcal{O} \neq \emptyset. \tag{14}$$

The robot is considered to be collision-free (probabilistically) if for all time $t \in [0, \tau]$ the probability for a collision to occur

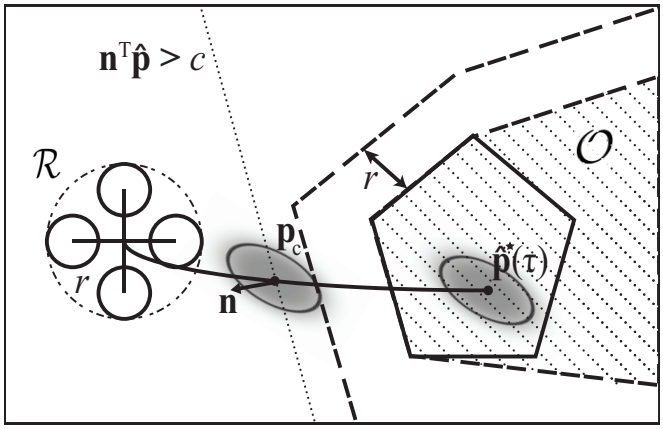

Fig. 2. Shown is a scenario where a quadrotor helicopter has uncertainty in both its trajectory estimation and the obstacle location. The a priori variance $P_c(\tau) + Z$ for a given obstacle normal $\mathbf{n}$ is represented as $\mathbf{n}^T(P_c(\tau) + Z)\mathbf{n}$. The collision point $\mathbf{p}_c$ is defined as the point along the trajectory where this transformed variance is some distance from the obstacle based on the selected confidence bound $\bar{p}$. The halfspace is defined such that $\mathbf{n}^T\hat{\mathbf{p}}(\tau, \Delta\mathbf{u}) > c$ where $c = \mathbf{n}^T\mathbf{p}_c$.

is less than the confidence bound $\bar{p}$ given the distributions of the position estimate and obstacle locations, i.e.,

$$p((\mathcal{R}(\mathbf{p}(\tau)) \cap \mathcal{O} \neq \emptyset) \mid (\mathcal{N}(\hat{\mathbf{p}}^{\star}(\tau), P_c(\tau) + Z) \cap \mathcal{O} \neq \emptyset)) \leq \bar{p}. \quad (15)$$

For a trajectory that is determined to be collision free, the current operator's input $\mathbf{u}$ is deemed safe and does not need to be changed, hence $\Delta\mathbf{u} = \mathbf{0}$. Conversely, if the probability for a collision is greater than the bound $\bar{p}$ then the operator's input is unsafe and must be corrected in order for the robot to obtain a collision-free trajectory, hence $\Delta\mathbf{u} \neq \mathbf{0}$.

Let $\mathbf{p}_c$ be defined as the first point along the trajectory that has a probability of colliding with an obstacle greater than the confidence bound, thus

$$\mathbf{p}_c = \hat{\mathbf{p}}^{\star}(t_c), \quad (16)$$

where

$$t_c = \underset{t\in[0,\tau]}{\operatorname{argmin}}\{ p((\mathcal{R}(\mathbf{p}(t)) \cap \mathcal{O} \neq \emptyset) \mid (\mathcal{N}(\hat{\mathbf{p}}^{\star}(\tau), P_c(\tau) + Z) \cap \mathcal{O} \neq \emptyset)) > \bar{p}\}. \quad (17)$$

The probabilities in Eqs. (15) and (17) can be difficult to compute exactly. Approximating the solution is desirable to make the algorithm tractable for real-time implementation.

Given a unit normal vector $\mathbf{n}$ of the obstacle $\mathcal{O}$ that points into the free workspace, consider a halfspace with the same normal $\mathbf{n}$ (pointing toward the free space) that provides a convex approximation of the local free space. The halfspace is located at the collision point $\mathbf{p}_c$, determined by Eq. (17).

Given the local approximation of the free space provided by the halfplane, the uncertainty can be mapped into the halfspace by transforming the multivariate distribution into a one-dimensional (along the normal $\mathbf{n}$) Gaussian distribution centered at $\mathbf{p_c}$,

$$\mathcal{N}(\hat{\mathbf{p}}^{\star}(\tau), (P_c(\tau) + Z)) \approx \mathcal{N}(\hat{\mathbf{p}}^{\star}(\tau), \mathbf{n}^T(P_c(\tau) + Z)\mathbf{n}). \quad (18)$$

Using this approximate representation of the uncertainty, the probability of avoiding collision can now be represented very simply by the number of standard deviations for a desired confidence bound.

Equation (15), given this approximate representation of the uncertainty, is now redefined such that the robot is considered to be collision-free for all time $t \in [0, \tau]$ if

$$\forall t \in [0, \tau] :: \mathcal{R}(\hat{\mathbf{p}}^{\star}(t)) \cap (\mathcal{O} \oplus \{\hat{\sigma}\mathbf{n}\}) = \emptyset, \quad (19)$$

where $\hat{\sigma}$ is the distance calculated from the standard deviation and selected confidence bound $\bar{p}$ where

$$\hat{\sigma} = a\mathbf{n}^T \left(\sqrt{P_c(\tau) + Z}\right)\mathbf{n}, \quad (20)$$

where a is a scaling factor that corresponds to a Chi-Squared distribution for the given confidence bound $\bar{p}$.

Equations (16) and (17) can now be approximated by the following simplified expression:

$$\mathbf{p}_c = \hat{\mathbf{p}}^{\star}\left(\underset{t\in[0,\tau]}{\operatorname{argmin}}\{\mathcal{R}(\hat{\mathbf{p}}^{\star}(t)) \cap (\mathcal{O} \oplus \{\hat{\sigma}\mathbf{n}\}) \neq \emptyset\}\right), \quad (21)$$

where if the system has no uncertainty $\hat{\sigma} = 0$, then Eqs. (19) and (21) are equivalent to the deterministic solution in [10].

Next, given Eqs. (16) and (19), a linear constraint is defined on the position $\hat{\mathbf{p}}(\tau, \Delta\mathbf{u})$ of the robot at time τ (see Fig. 2)

$$\mathbf{n}^T\hat{\mathbf{p}}(\tau, \Delta\mathbf{u}) > \mathbf{n}^T\mathbf{p}_c. \quad (22)$$

Substituting Eq. (11) from Assumption 3, the constraint on the robot's position in Eq. (22) can be transformed into a constraint on its change in input $\Delta\mathbf{u}$

$$\mathbf{n}^T J(\tau)\Delta\mathbf{u} > \mathbf{n}^T(\mathbf{p}_c - \hat{\mathbf{p}}^{\star}(\tau)). \quad (23)$$

Equation (9) is approximated using Eq. (23) as

$$\begin{aligned} &\text{minimize: } \Delta\mathbf{u}^T R\Delta\mathbf{u} \\ &\text{subject to: } \mathbf{n}^T J(\tau)\Delta\mathbf{u} > \mathbf{n}^T(\mathbf{p}_c - \hat{\mathbf{p}}^{\star}(\tau)), \end{aligned} \quad (24)$$

where solving this convex optimization, such as is done by the RVO library in [29], provides a collision free change in input $\Delta\mathbf{u}$ with the control input given to the robot as $\mathbf{u}+\Delta\mathbf{u}$.

B. Handling Convex Edges and Corners Through Iteration

The use of an approximation of a convex region of the local free space near the robot's trajectory means that it cannot be assumed that the newly selected control input $\mathbf{u}+\Delta\mathbf{u}$ avoids collisions with respect to *all* obstacles for all time $t \in [0, \tau]$. This is, in particular, true near convex edges or corners of the workspace as shown in Fig. 3. However, the approach can simply be repeated in an iterative fashion to solve this problem.

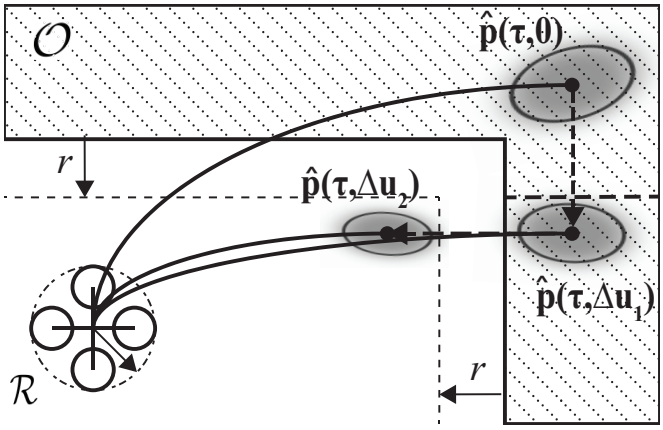

Fig. 3. The iterative process of the algorithm is shown. A trajectory is estimated from the original operator's input such that $\Delta\mathbf{u} = \mathbf{0}$ and is shown as $\hat{\mathbf{p}}(\tau, \mathbf{0})$. The variance on this estimated position is shown as the gray ellipsoid located at $\hat{\mathbf{p}}(\tau, \mathbf{0})$. Considering this uncertainty, a new input is determined $\mathbf{u} + \Delta\mathbf{u}_1$ to avoid the first detected collision. The trajectory for the new input is predicted and is shown with its variance at $\hat{\mathbf{p}}(\tau, \Delta\mathbf{u}_1)$. This also results in a collision and the algorithm computes a new change in input $\Delta\mathbf{u}_2$. The resulting trajectory $\hat{\mathbf{p}}(\tau, \Delta\mathbf{u}_2)$ is collision free and the input $\mathbf{u} + \Delta\mathbf{u}_2$ is passed to the robot.

Assume the algorithm has computed a change in control input according to Eq. (9) and that it is the first iteration of the algorithm. Continuing to iteration i, the control input $\mathbf{u} + \Delta\mathbf{u}_i$ is used to extrapolate the trajectory and check for a potential collision. If a collision is found to occur, a new linear constraint is defined as

$$\mathbf{a}_i^T \Delta\mathbf{u} > b_i, \tag{25}$$

where

$$\mathbf{a}_i^T = \mathbf{n}_i^T J_i(\tau), \tag{26}$$

$$b_i = \mathbf{n}_i^T (\mathbf{p}_{c,i} - \hat{\mathbf{p}}_i(\tau)). \tag{27}$$

The convex optimization problem in Eq. (24) can now be solved for all iterations i by the following:

$$\begin{aligned} &\text{minimize:} && \Delta\mathbf{u}^T R \Delta\mathbf{u} \\ &\text{subject to:} && \textstyle\bigcap_{j=1}^{i} \{\mathbf{a}_j^T \Delta\mathbf{u} > b_j\}. \end{aligned} \tag{28}$$

Every i^{th} iteration of the algorithm introduces an additional constraint to the convex optimization problem. After at most d iterations, the control input $\mathbf{u} + \Delta\mathbf{u}$, where $\Delta\mathbf{u}$ is the change in control input computed in the latest iteration, is then applied to the robot. The number of iterations, and therefore, the number of constraints is maximized at d, where d is the dimension of the workspace. This upper limit accounts for corners of the free space in d dimensions as shown in Fig. 3. This iterative approach is performed during every sensing-action cycle of the robot.

This iterative approach aligns with the LP-type algorithm in [30]. The LP-algorithm solves low-dimensional convex optimization problems in $O(i)$ expected time by considering the constraints in an iterative fashion, where i is the number of constraints. The dimension of the optimization problem in this paper equals the dimension n of the control input $\Delta\mathbf{u}$, which, typically, is equal to the dimension d of the workspace. Maximizing the number of iterations to d ensures the convex optimization problem remains feasible.

V. Simulation: Results and Discussion

The proposed approach is implemented in simulation on a quadrotor helicopter. Results and discussion are presented below.

A. Implementation Details

All computations were performed on a desktop computer with an Intel Core i7-4790K, 8GB RAM, and the 64-bit Ubuntu 12.04 operating system. The algorithm was implemented within the Robot Operating System (ROS) framework [31]. The simulations used a control cycle frequency of 50 Hz. The VRep simulator from Coppelia Robotics [32] was used to simulate the behavior of the quadcopter. The quadcopter was controlled through the use of the V-Rep ROS plugin that allows communication between a running ROS node and the simulator. The V-Rep simulator sends the position of the robot into ROS while the ROS node sends the control input to be applied to the robot.

The obstacles in the environment are predefined for each simulation scene and represented as oriented triangular facets. These triangles model the true obstacles offset along their normals by the radius r of the bounding sphere of the robot, approximating the Minkowski difference of the robot and the obstacles so the robot can be considered as a point. The trajectory of the robot is estimated by integrating Eq. (1) forward in time using a Runge-Kutta integration with 0.01s time-steps. Each increment of the trajectory is considered a straight-line segment that is checked for intersection with the obstacle's triangular facets. The matrices $J(\tau)$ and $L(\tau)$ were approximated through numerical differentiation.

1) Quadcopter Dynamics: The simulations incorporated a model of a quadrotor helicopter similar to work in [10]. The model has a 12-dimensional state $\mathbf{x} = [\mathbf{p}^T, \mathbf{v}^T, \mathbf{r}^T, \mathbf{w}^T]^T \in \mathcal{X}$ that consists of position $\mathbf{p} \in \mathbb{R}^3$, velocity $\mathbf{v} \in \mathbb{R}^3$, orientation $\mathbf{r} \in \mathbb{R}^3$ (rotation about $\mathbf{r}/\|\mathbf{r}\|$ by an angle $\|\mathbf{r}\|$), and angular velocity $\mathbf{w} \in \mathbb{R}^3$. The 3-dimensional control input $\mathbf{u} = [u_z, u_r, u_p]^T \in \mathcal{U}$ consists of the desired vertical velocity u_z, desired roll u_r, and desired pitch u_p. Typically, a quadcopter also has input for the yaw, but this is a redundant degree-of-freedom that is held fixed at zero. The equations of motion are given as

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{29}$$

$$\dot{\mathbf{v}} = -k_{\text{drag}}\mathbf{v} + exp([\mathbf{r}])[0, 0, k_{p1}(u_z - v_z)]^T, \tag{30}$$

$$\dot{\mathbf{r}} = \mathbf{w}, \tag{31}$$

$$\dot{\mathbf{w}} = \begin{bmatrix} k_{p2}(u_r - r_x) - k_d w_x \\ k_{p2}(u_p - r_y) - k_d w_y \\ -k_{p3} w_z \end{bmatrix}, \tag{32}$$

where $[\mathbf{r}]$ represents the skew-symmetric cross product matrix of $\mathbf{r}$. The terms $k_{\text{drag}} = 0.2, k_d = 0.1, k_{p1} = 1.0, k_{p2} = 10.0, k_{p3} = 0.1$, and $k_{p4} = 0.05$ are coefficients and gains

whose values were estimated to provide realistic motion. The robot was simulated with an aggressive flight model that allows for maximum roll and pitch angles of 0.35 rad (20 degrees).

2) Artificial Uncertainty: Artificial noise was included in the simulations to represent the uncertainty in the trajectory estimate and the obstacle locations. For the uncertainty in the estimated trajectory, as the robot propagates its state forward in time according to the dynamics, a sample was drawn from a normal distribution of variance M and added according to Eq. (1). For the uncertainty in the obstacle location, a random sample was drawn from a normal distribution of variance Z at the beginning of each sensing-action cycle. This value was then added to the relative obstacle location that is newly provided to the robot every sensing-action cycle.

The value to offset the halfplane based on the confidence bound $\hat{\sigma}$ was calculated by first taking taking the Cholesky decomposition of the covariance matrix such that $\Sigma = \sqrt{\Sigma}\sqrt{\Sigma}^T$. The matrix $\sqrt{\Sigma}$ was then projected onto the normal and scaled as $a\mathbf{n}\sqrt{\Sigma}\mathbf{n}^T$ where a represents the value from the Chi-Squared distribution for a given confidence bound. For the experiments discussed in the subsequent section, the confidence bound was set to be $\bar{p} = 0.5$, or in other words the robot has a probability of *not* colliding that is at least 95% and, therefore, $a = 3.841$.

B. Results and Discussion

Experiments were performed to investigate the relationship between the confidence bound $\bar{p}$ and the true probability of avoiding collisions using the algorithm presented in this paper. These experiments were performed by placing the quadcopter directly next to a wall and providing the robot with a constant control input in the direction of the wall, or in other words opposite the wall's normal $\mathbf{n}$. This constant input and initial position constantly provides a potential collision. For this experiment, the motion covariance and sensing covariance were defined as

$$M = \text{diag}([20^2 I_3,\ 10^2 I_3,\ 5^2 I_3,\ 2.5^2 I_3])10^{-4}, \quad (33)$$

$$Z = 0.2^2 I_3, \quad (34)$$

where $\text{diag}(\ldots)$ represents a block-diagonal matrix, I_3 represents a 3×3 identity matrix. The experiment consisted of $200,000$ time steps and the number of collisions was recorded. The true probability of collision for the algorithm was found to be 0.713% with an accurate estimate of the covariance. This is significantly less probable to have a collision than the 5% defined by $\bar{p}$ due to the conservative nature of the algorithm.

The covariance values can, in practice, be difficult to properly estimate, particularly the covariance with respect to the robot's model. The sensitivity of the algorithm with respect to erroneous values in covariance estimates was tested. An experiment was performed where the true uncertainty applied to the robot model and the obstacle location, M and Z respectively, are underestimated by the algorithm when computing a change in input [Eq. (20)] by 25% and 50%. At a 25% underestimate of the covariance, the true probability of collision only increased by 24.3% to a value of 0.886%. For an underestimate of the covariance by 50%, the true probability of collision increased by 176% to 1.97%. This shows that the true probability of collision is sensitive to the errors in covariance estimates, but due to the conservative nature of the algorithm, it is still robust to errors in the covariance estimates.

TABLE I
ERROR CALCULATIONS AS FRACTION OF RADIUS

	Smaller Covariance		**Larger Covariance**	
	x	y	x	y
Maximum Deviation	10.33	10.01	10.28	8.231
RMS-Average	4.811	5.317	4.634	4.240
Standard Deviation	4.813	3.602	4.621	2.507

Next, three experiments were performed to evaluate the deviation from the nominal (deterministic) path in the stochastic approach. In these experiments, the robot was controlled with a constant input in the negative $\mathbf{y}$-direction (see Fig. 4) for a fixed amount of time. The first experiment ran the deterministic version of the algorithm. The second and third experiments ran the stochastic version of the algorithm as developed in this paper. The second experiment used a smaller covariance of

$$M = \text{diag}([5^2 I_3,\ 2.5^2 I_3,\ 1.25^2 I_3,\ 0.625^2 I_3])10^{-4}, \quad (35)$$

$$Z = 0.05^2 I_3, \quad (36)$$

while the third experiment used a larger covariance of

$$M = \text{diag}([10^2 I_3,\ 5^2 I_3,\ 2.5^2 I_3,\ 1.25^2 I_3])10^{-4}, \quad (37)$$

$$Z = 0.1^2 I_3. \quad (38)$$

The entire trajectories were recorded during the experiments and are shown in Fig. 4. The deviations of trajectories for both covariance values were calculated and are presented in Fig. 4. At every time-step, the deviation in the trajectory was calculated in the $x - y$ plane. The deviation in the $\mathbf{z}$ direction is negligible because the algorithm does not change the input with respect to the robot's altitude due to the obstacles being vertically aligned. In the $\mathbf{x}$ and $\mathbf{y}$ directions, the maximum deviation over the entire trajectory was calculated as well as the RMS-average value and the standard deviation. The results of those calculations, normalized by the robot's radius, are given in Table I for both covariance values. As can be seen, the algorithm can have large deviations from the deterministic results in the presence of uncertainty while still avoiding collisions. The maximum deviations were observed to correlate with the robot taking a more conservative trajectory around the ends of the obstacles and the deviation grew over time as the deterministic case results in a faster completion of this trajectory.

A simulation was performed where the quadcopter was guided through a window-like opening in a large wall (see Fig. 5). The goal position of the robot was set directly on the other side of the window from the quadcopter's initial

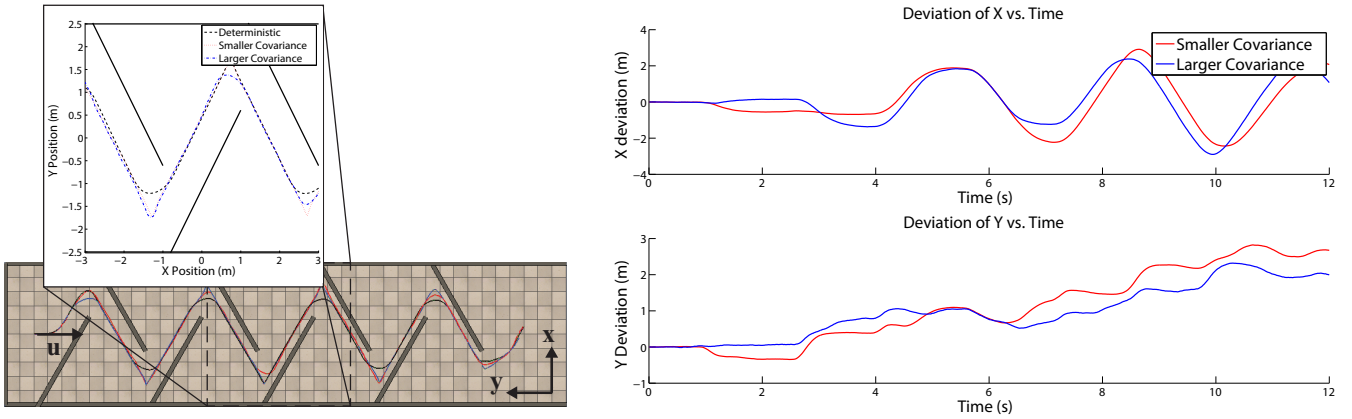

Fig. 4. An experiment was performed to compare the path of the quadcopter through the environment for deterministic collision avoidance and stochastic collision avoidance with two sets of variances [Eqs. (35)–(38)]. On the left, the resulting paths are given and a zoomed portion is given for clarity. On the right, the deviations from the deterministic case are given for the two covariances.

position. The quadcopter strafes along the non-vertical wall and then passes through the window when it reaches it and is not avoiding collisions with the wall in front of it. This simulation demonstrates the algorithms capabilities to perform 3-d collision avoidance with non-vertical obstacles even when using a simple 1-d approximation of the uncertainty.

Videos of the above experiments along with other scenarios can be found at the University of Utah DARC Lab webpage[1].

VI. Conclusions and Future Work

A feedforward-based collision avoidance algorithm for tele-operated unmanned aerial vehicles that explicitly considers uncertainty was presented. Specifically, the method estimates the trajectory of the robot from the current time into the future for some predetermined amount of time given the robot's dynamics and the control input (from the operator). This trajectory is checked for any collisions with the obstacles in the environment given the uncertainty in the estimated trajectory and the uncertainty in the obstacle location. Experiments were performed on a simulated quadrotor helicopter that showed the approach is capable of avoiding collisions with a probability greater than a selected confidence bound. The approach provides an input that is as close as possible to the original operator's input while avoiding collisions. The minimal change in user input provides a method to control the quadcopter that is intuitive and safe, allowing the operator to focus on other tasks.

The approach was developed for general, nonlinear dynamics. Future work includes implementation on different types of mobile robotic systems and on-line implementation involving on-board range-finding sensors, such as LIDAR. Additionally, authors plan to apply the technology to UAVs for search and rescue, to enhance situational awareness for first responders, and to enable autonomous environmental monitoring in urban environments.

[1]http://www.kam.k.leang.com/academics/robotics/

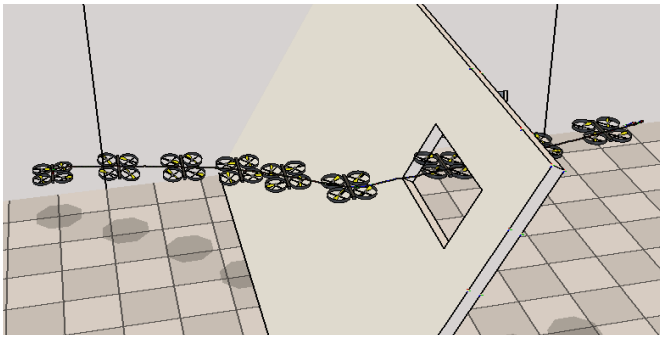

Fig. 5. A 3-dimensional example is shown where the quadrotor is steered towards a goal point through a window on a slanted wall. The window has tight clearance with respect to the robot. The height of the window is only 25% larger than the robot's diameter, however, the diameter is a conservative estimate provided by the minimum-radius bounding sphere. The width of the window is 75% larger than the robot's diameter.

VII. Acknowledgments

This material is based upon work supported by the National Science Foundation, Partnership for Innovation Program, Grant No. 1430328. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] D. Hausamann, W. Zirnig, G. Schreier, and P. Strobl, "Monitoring of gas pipelines a civil UAV application," *Aircraft Engineering and Aerospace Technology*, vol. 77, no. 5, pp. 352 – 360, 2005.

[2] F. Nex and F. Remondino, "UAV for 3D mapping applications: a review," *Applied Geomatics*, vol. 6, no. 1, pp. 1 – 15, 2013.

[3] T. Brown, S. Doshi, S. Jadhav, and J. Himmelstein, "Test bed for a wireless network on small UAVs," in *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, vol. AIAA 2004-6480, 2004.

[4] M. Lega and R. M. A. Napoli, "Aerial infrared thermography in the surface waters contamination monitoring," *Desalination and Water Treatment*, vol. 23, no. 1-3, 2010.

[5] H. S. Trammell, A. R. Perry, S. Kumar, P. V. Czipott, B. R. Whitecotton, T. J. McManus, and D. O. Walsh, "Using unmanned aerial vehicle-borne magnetic sensors to detect and locate improvised explosive

devices and unexploded ordnance," in *Proc. SPIE Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IV*, vol. 5778, 2005.

[6] P. S. Lin, L. Hagen, K. Valavanis, and H. Zhou, "Vision of unmanned aerial vehicle (UAV) based traffic management for incidents and emergencies," in *12th World Congress on Intelligent Transport Systems*, 2005.

[7] M. Neri, A. Campi, R. Suffritti, F. Grimaccia, P. Sinogas, O. Guye, C. Papin, T. Michalareas, L. Gazdag, and I. Rakkolainen, "SkyMedia - UAV-based capturing of HD/3D content with WSN augmentation for immersive media experiences," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2011.

[8] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[9] S. Omari, P. Goh, M. Burri, M. Achtelik, and R. Siegwart, "Visual industrial inspection using aerial robots," in *Applied robotics for the Power Industry*, 2014.

[10] J. Israelsen, M. Beall, D. Bareiss, D. Stuart, E. Keeney, and J. van den Berg, "Automatic collision avoidance for manually tele-operated unmanned aerial vehicles," in *IEEE Int. Conf. on Robotics and Automation*, 2014.

[11] J. Byrne, M. Cosgrove, and R. Mehra, "Stereo based obstacle detection for an unmanned air vehicle," in *IEEE Int. Conf. on Robotics and Automation*, 2006.

[12] W. P. Wang, "Three-dimensional collision avoidance in production automation," *Computers in Industry*, vol. 15, no. 3, pp. 169 – 174, 1990.

[13] S. D'Attanasio, O. Tonet, G. Megali, M. C. Carrozza, and P. Dario, "A semi-automatic handheld mechatronic endoscope with collision-avoidance capabilities," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1586 – 1591.

[14] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 7, pp. 278–288, Jun 1991.

[15] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[16] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 19, pp. 1179–1187, 1989.

[17] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[18] P. Fiorini and Z. Shiller, "Motion planning in dynamic environmnts using velocity obstacles," *Int. Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[19] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 723–730, May 2011.

[20] S. Patil, J. van den Berg, and R. Alterovitz, "Estimating probability of collision for safe planning under gaussian motion and sensing uncertinaty," in *IEEE Int. Conf. on Robotics and Automation*, 2012.

[21] P. Missiuro and N. Roy, "Adapting probabilistic roadmaps to handle uncertain maps," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1261–1267, May 2006.

[22] J. Müller and G. Sukhatme, "Risk-aware trajectory generation with application to safe quadrotor landing," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2014.

[23] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *IEEE Int. Conf. on Robotics and Automation*, (Shanghai, China), 2011.

[24] S. Li and G. Tao, "Feedback based adaptive compensation of control system sensor uncertainties," *Automatica*, vol. 45, no. 2, pp. 393 – 404, 2009.

[25] M. Adams, W. S. Wijesoma, and A. Shacklock, "Autonomous navigation: achievements in complex environments," *IEEE Instrum. Meas. Mag.*, vol. 10, no. 3, pp. 15 – 21, 2007.

[26] E. J. Rodriguez-Seda, D. M. Stipanovic, and M. W. Spong, "Collision avoidance control with sensing uncertainties," in *American Control Conference*, pp. 3363 – 3368, 2011.

[27] H. Everett, "Survey of collision avoidance and ranging sensors for mobile robots," *Robotics and Autonomous Systems*, vol. 5, no. 1, pp. 5 – 67, 1989.

[28] J. P. Chryssanthacopoulos and M. J. Kochenderfer, "Accounting for state uncertainty in collision avoidance," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 951 – 960, 2011.

[29] J. van den Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Proc. Int. Symp. on Robotics Research*, pp. 3–19, 2011.

[30] M. De Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[31] M. Quigley, K. Conley, B. Gerkey, J. Faust, and T. Foote, "Ros: An open-source robot operating system," in *IEEE Int. Conf. on Robotics and Automation: Workshop on Open-Source Software*, 2009.

[32] E. Rohmer, S. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *IEEE Int. Conf. on Intelligen Robots and Systems*, 2013.