Auton Robot
DOI 10.1007/s10514-017-9614-4

On-board model-based automatic collision avoidance: application in remotely-piloted unmanned aerial vehicles

Daman Bareiss[1] · Joseph R. Bourne[1] · Kam K. Leang[1]

Received: 2 May 2016 / Accepted: 11 January 2017

Abstract This paper focuses on real-world implementation and verification of a local, model-based stochastic automatic collision avoidance algorithm, with application in remotely-piloted (tele-operated) unmanned aerial vehicles (UAVs). Automatic collision detection and avoidance for tele-operated UAVs can reduce the workload of pilots to allow them to focus on the task at hand, such as searching for victims in a search and rescue scenario following a natural disaster. The proposed algorithm takes the pilot's input and exploits the robot's dynamics to predict the robot's trajectory for determining whether a collision will occur. Using on-board sensors for obstacle detection, if a collision is imminent, the algorithm modifies the pilot's input to avoid the collision while attempting to maintain the pilot's intent. The algorithm is implemented using a low-cost on-board computer, flight-control system, and a two-dimensional laser illuminated detection and ranging sensor for obstacle detection along the trajectory of the robot. The sensor data is processed using a split-and-merge segmentation algorithm and an approximate Minkowski difference. Results from flight tests demonstrate the algorithm's capabilities for tele-operated collision-free control of an experimental UAV.

Keywords Collision avoidance · Control · Aerial robots

✉ Kam K. Leang
kam.k.leang@utah.edu

Daman Bareiss
daman.bareiss@utah.edu

Joseph R. Bourne
joseph.bourne@utah.edu

1 Design, Automation, Robotics, and Control (DARC) Laboratory, University of Utah Robotics Center, Department of Mechanical Engineering, University of Utah, Salt Lake City, UT 84112, USA

1 Introduction

Recent advances in unmanned aerial vehicles (UAVs), such as improved flight times (Abdilla et al. 2015; Gatti et al. 2015), advanced flight-control systems (Mahony et al. 2012), and reduced development costs (Han et al. 2013), have led to a dramatic increase in the number of civil and commercial applications for UAVs. Applications of UAV technology include mapping and media resources (Nex and Remondino 2013; Neri et al. 2011), search and rescue (Waharte and Trigoni 2010), precision farming (Barrientos et al. 2011), space exploration (Landis 2004), traffic management (Lin 2005), environmental monitoring (Hausamann et al. 2005; Trammell et al. 2005), telecommunication (Brown et al. 2004), and even for entertainment (Yoshimoto et al. 2009). In fact, many small multirotor UAVs (such as quadcopters) have the ability to access indoor locations or complex urban environments that may be hard to reach or unsafe for humans. These UAVs are ideally suited for search and rescue, law enforcement, and/or emergency response to enhance situational awareness (Goodrich et al. 2008; Tomic et al. 2012; Cole et al. 2006; Cook et al. 2015; Kumar et al. 2011).

Despite recent advances in the design and development of UAVs, particularly hover-capable rotorcraft UAVs, the task of carefully controlling the UAV through a cluttered environment and avoiding a collision with nearby obstacles and humans remains a challenge (Mohammed et al. 2014; Valavanis et al. 2014). Thus, one of the most daunting tasks for even a skilled UAV pilot is collision avoidance, especially when a UAV is deployed to help look for survivors inside of a partially collapsed building where usually the only feedback information is a live-camera feed from the vehicle. The need for automatic collision avoidance technology is critical and necessary to allow pilots to focus on higher-priority tasks such as locating survivors and/or assessing potential dangers and damage.

Herein, an on-board model-based automatic collision avoidance algorithm that considers sensing and estimation uncertainties while maintaining the user's intent is described, implemented, and validated on a custom-designed experimental multi-rotor UAV system. 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. This work is based on leveraging the theoretical developments presented by (Israelsen et al. 2014; Bareiss et al. 2015) and it is not only applicable to UAVs, but the algorithm can be applied to other robotic and autonomous systems – such as self-driving cars – where collision avoidance is needed.

The block diagram of the collision avoidance approach along with the new proposed on-board sensing scheme is shown in Fig. 1. During flight, the UAV with on-board computation continuously predicts the trajectory of the robot given the pilot's input. At the same time, on-board sensing such as laser illuminated detection and ranging (LIDAR) sensors are used for sensing obstacles along the trajectory of the robot. To allow implementation on low-cost on-board computation, the LIDAR data is processed using a split-and-merge segmentation algorithm and an approximate Minkowski difference, and the information is used to predict a collision. If the predicted trajectory and the sensing information result in a possible collision, then the algorithm alters the input to the robot to avoid a collision. Measurement error combined with uncertainty in state estimation (due to the fact that a model of the robot's dynamics are used) are also considered in the algorithm. It is pointed out that when on-board sensing technology is used the uncertainty in the sensor's output can 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. Because sensing and state estimation uncertainties are inherent in real-world applications, it is necessary to consider these uncertainties in the collision avoidance algorithm.

The main contribution of this work is the real-world implementation and verification of the proposed local, model-based automatic collision avoidance algorithm for remotely-piloted UAVs with on-board sensing. Compared to existing local or reactive approaches such as the potential field technique (Khatib 1986), the dynamic window approach (Fox et al. 1997), velocity obstacles (Fiorini and Shiller 1998), and vector field histograms (VFH) (Borenstein and Koren 1991), the proposed approach is based upon local sensor information but can exploit both global information as well. The approach also considers the uncertainty in the

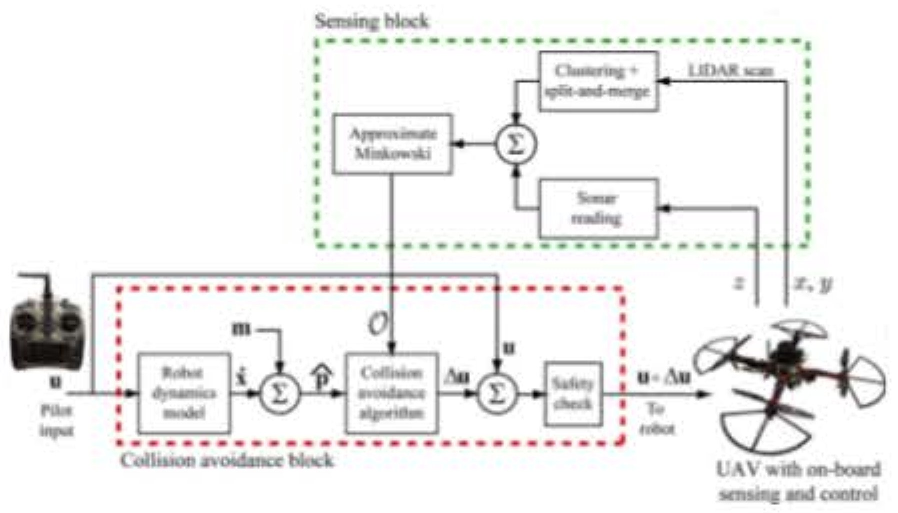


Fig. 1 Automatic collision avoidance for tele-operated UAVs: A UAV pilot controls the aerial vehicle and provides input $\mathbf{u}$. When a collision is detected with on-board sensing and state estimation, the algorithm produces an input $\Delta\mathbf{u}$ that augments the pilot's input $\mathbf{u}$ to steer the robot away from the obstacle. The control block diagram includes the sensing block and 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 $\mathcal{O}$. 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

estimation and measurement process, and applies to the full (possibly nonlinear) robot dynamics.

The remainder of this paper is structured as follows. A detailed summary of related work and the comparison of this work to similar techniques is presented in Sect. 2. A detailed summary of the stochastic collision avoidance algorithm is given in Sect. 3, followed by a description of the custom-designed experimental UAV system with on-board computation and obstacle detection in Sect. 4. The experimental results and discussion are presented in Sect. 5. Finally, concluding remarks and a discussion of future work are presented in Sect. 6.

2 Related work and state-of-the-art

Collision avoidance is an important research topic in robotics, where numerous approaches have been developed and applied to manufacturing systems (Wang 1990), medical devices (D'Attanasio et al. 2000), and mobile service robots (Borenstein and Koren 1991). Some early methods include potential fields (Khatib 1986), the dynamic window approach (Fox et al. 1997), velocity obstacles (Fiorini and Shiller 1998), and vector field histograms (VFH) (Borenstein and Koren 1991). In general, collision avoidance methods can be classified into one of two main categories: global (motion planning methods) and local (reactive methods).

First, avoiding collisions between a mobile robot and obstacles can be achieved through a global motion planning algorithm which typically assumes *a priori* information about the environment (Niewenhuisen and Behnke 2015;

Patil et al. 2012; Muller and Sukhatme 2014). In (Achtelik et al. 2014), a path planning approach is used where collisions are avoided during the trajectory generation stage. 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 while reaching some desired goal state (i.e., position). Often, global planners are computationally expensive and complete information about the environment is required. Thus, in applications where robots are equipped with cameras for search and rescue in collapsed buildings or in unstructured environments, global planners may provide limited performance.

The 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 some local knowledge (sensor information) of the obstacles and environment. Many of the reactive algorithms use relative sensing information and develop control laws that guarantee separation between agents (and obstacles) in the presence of uncertainty (Rodriguez-Seda et al. 2011). Other techniques deal with state uncertainty by exploiting dynamic programming (Chryssanthacopoulos and Kochenderfer 2011).

In many implementations of the local algorithms the collision avoidance is approximated through a first-order model by predicting time to impact between the robot and an obstacle while only considering some maximum acceleration (Mendes and Ventura 2013; Brand et al. 2014; Rehmtullah and Kelly 2015; Stegagno et al. 2014). The algorithm in this paper performs collision avoidance with an explicit model of the robot's full, possibly nonlinear, equations of motion, rather than approximating collisions through the relative velocity formulation.

Integrated global and local planners have been explored, where these algorithms use a predicted trajectory to avoid collisions with the observable, local obstacle (Fiorini and Shiller 1998). Typically, these algorithms avoid collisions by computing a given change in input for a current sensing-action cycle (Li and Tao 2009; Adams et al. 2007; Rodriguez-Seda et al. 2011). The approach in this paper is intended for use with local sensor information, however, it can also be applied in situations where global knowledge is provided. Additionally, the formulation is applicable to the full robot dynamics and the method considers explicitly the uncertainty in the estimation and measurement process.

In (Mendes and Ventura 2013), the FastSLAM algorithm (Montemerlo et al. 2002) is implemented to predict distance to obstacles. Much research has been focused on using vision to detect and avoid obstacles, especially in the field of autonomous automobiles (Huh et al. 2008; Bernini et al. 2014). Vision has been applied to obstacle detection and avoidance in UAVs as well (Agrawal et al. 2015; Matthies et al. 2014; Saha et al. 2014; Mejias et al. 2010). Vision has been shown to provide robust obstacle position estimates, but it can be computationally expensive when compared to the more traditional approach of using range-based sensors such as described in (Astilla et al. 2015; Maier et al. 2012; Wang et al. 2015).

3 Automatic collision avoidance algorithm

This section presents the details and main results of the model-based stochastic automatic collision avoidance algorithm. Additional details of the theoretical framework are described in (Bareiss et al. 2015). First, the problem is formally defined below in Sect. 3.1, followed by details of the algorithm in Sect. 3.2. It is pointed out that the work in (Bareiss et al. 2015; Israelsen et al. 2014) assumed that the UAV was not able to yaw by operator commands. On the other hand, for many applications including search-and-rescue, the UAV pilot must be able to provide a yaw command to the robot to enable the operator to search and survey a given area through video feedback. Thus, the method proposed herein incorporates yaw for practical application in UAVs and Sect. 3.2.4 presents the details.

3.1 Problem formulation

3.1.1 Notation

Throughout this paper, vectors are denoted by boldface lower-case letters, for example **a**. Vector sets are represented by calligraphics, such as $\mathcal{A}$. Scalar and matrices are denoted by lowercase italic letters, such as a, and uppercase italic letters, such as A, respectively. Scalar and matrix multiplications as well as Minkowski sums of sets are defined as follows:

$$x\mathcal{A} = \{x\mathbf{a} \mid \mathbf{a} \in \mathcal{A}\}, \tag{1}$$

$$X\mathcal{A} = \{X\mathbf{a} \mid \mathbf{a} \in \mathcal{A}\}, \tag{2}$$

$$\mathcal{X} \oplus \mathcal{A} = \{\mathbf{x} + \mathbf{a} \mid \mathbf{x} \in \mathcal{X}, \mathbf{a} \in \mathcal{A}\}. \tag{3}$$

A vector **a** that is sampled from a normal distribution with mean $\bar{\mathbf{a}}$ and variance Σ, where Σ is positive-semidefinite, is given by

$$\mathbf{a} \sim \mathcal{N}(\bar{\mathbf{a}}, \Sigma). \tag{4}$$

3.1.2 Problem definition

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

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{m}, \quad \mathbf{m} \sim \mathcal{N}(\mathbf{0}, M), \tag{5}$$

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 the relationship

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\mathbf{x}(\hat{0}), \mathbf{u}, t). \tag{6}$$

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

$$\mathbf{x}(t) \sim \mathcal{N}(\hat{\mathbf{x}}(t), P(t)), \tag{7}$$

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 Eq. (5), and $P(t)$ is the uncertainty of the state, defined as

$$P(t) = \mathrm{E}\left[(\mathbf{x}(t) - \hat{\mathbf{x}}(t))(\mathbf{x}(t) - \hat{\mathbf{x}}(t))^T\right]. \tag{8}$$

The uncertainty at time t, Eq. (8), is found by solving the following differential equation:

$$\dot{P}(t) = A(t)P(t) + P(t)A(t)^T + M, \tag{9}$$

$$A(t) = \frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}}(\hat{\mathbf{x}}(t), \mathbf{u}(t)). \tag{10}$$

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, meaning the subspace of the workspace that cannot be seen by the robot is also considered an obstacle. The obstacles' positions can be defined relative to the robot's position including uncertainty with known variance $Z(\mathbf{x}) \in \mathbb{R}^{d\times d}$ as

$$\mathcal{O} = \bigcup_{i=1}^{n} \mathcal{O}_i \oplus \{\mathbf{w}\}, \quad \mathbf{w} \sim \mathcal{N}(0, Z(\mathbf{x})), \tag{11}$$

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 $Z(\mathbf{x})$. The variance in the sensing $Z(\mathbf{x})$ is a function of the state $\mathbf{x}$ to represent how the uncertainty in some sensors can change with the distance to obstacles, such as with laser rangefinders the uncertainty typically decreases as the distance to an object decreases.

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

$$\mathcal{R}(\mathbf{x}(t)) \cap \mathcal{O} \neq \emptyset. \tag{12}$$

The problem is now 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 some 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 $\bar{p}$ for all time less than the time horizon $\tau \in \mathbb{R}$, therefore

$$\begin{aligned} &\text{minimize: } \Delta\mathbf{u}^T R \Delta\mathbf{u} \\ &\text{subject to:} \\ &\quad p\left(\forall t \in [0, \tau]{::}\mathcal{R}(\mathbf{x}(t)) \cap \mathcal{O} = \emptyset \mid P(t), Z(\mathbf{x})\right) \leq \bar{p}, \end{aligned} \tag{13}$$

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

3.2 Technical approach

In the following, a model-based feedforward approach is presented for collision avoidance (see block diagram in Fig. 1).

3.2.1 Assumptions

First, the following assumptions are made to simplify the nonlinear, non-convex optimization problem for real-time implementation with potentially limited computational power:

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

$$\mathbf{p}(t) = C\mathbf{x}(t), \tag{14}$$

where $C \in \mathbb{R}^{d\times m}$.
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, or ellipsoidal, subset of the workspace occupied by the robot at position $\mathbf{p}$.
3. The robot's trajectory can be represented through a first-order Taylor expansion, i.e.,

$$\hat{\mathbf{p}}(t, \Delta\mathbf{u}) \approx \hat{\mathbf{p}}^\star(t) + J(t)\Delta\mathbf{u}, \tag{15}$$

where

$$\hat{\mathbf{p}}^\star(t) = C\mathbf{g}(\hat{\mathbf{x}}, \mathbf{u}, t), \;\; J(t) = C\frac{\partial \mathbf{g}}{\partial \mathbf{u}}(\hat{\mathbf{x}}, \mathbf{u}, t). \tag{16}$$

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.

3.2.2 Algorithmic solution

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. (15) (see block diagram in Fig. 1). The variance on the predicted state is given in Eq. (8).

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

$$P_c(t) = C P(t) C^T. \tag{17}$$

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

$$\mathcal{P}_{\text{collision}} = (\mathcal{N}(\hat{\mathbf{p}}^\star(\tau), P_c(\tau) + Z(\mathbf{x})) \cap \mathcal{O}) \neq \emptyset. \tag{18}$$

The robot is considered to be probabilistically collision-free for all time $t \in [0, \tau]$ if the probability for a collision to occur is less than the confidence bound $\bar{p}$ given the distributions of the position estimate and obstacle locations, i.e.,

$$p((\mathcal{R}(\mathbf{p}(\tau)) \cap \mathcal{O} \neq \emptyset) \mid (\mathcal{P}_{\text{collision}} \neq \emptyset)) \leq \bar{p}. \tag{19}$$

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 $\Delta\mathbf{u} = \mathbf{0}$. However, if the probability for a collision is greater than the confidence bound $\bar{p}$ then the operator's input is deemed unsafe and must be corrected in order for the robot to obtain a collision-free trajectory, resulting in $\Delta\mathbf{u} \neq \mathbf{0}$.

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

$$\mathbf{p}_c = \hat{\mathbf{p}}^\star(t_c), \tag{20}$$

where

$$t_c = \operatorname{argmin}_{t\in[0,\tau]}\{p((\mathcal{R}(\mathbf{p}(t)) \cap \mathcal{O} \neq \emptyset) \mid (\mathcal{P}_{\text{collision}} \neq \emptyset)) > \bar{p}\}. \tag{21}$$

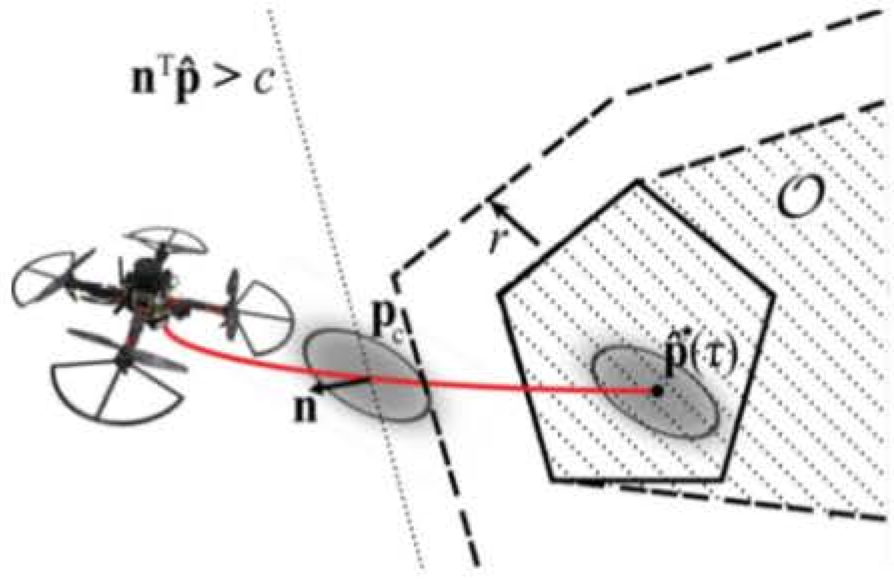


Fig. 2 Shown is a scenario where a quadcopter UAV has uncertainty in both its trajectory estimation and the obstacle location. The a priori variance $P_c(\tau) + Z(\mathbf{x})$ for a given obstacle normal $\mathbf{n}$ is represented as $\mathbf{n}^T(P_c(\tau) + Z(\mathbf{x}))\mathbf{n}$. The collision 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 $c = \mathbf{n}^T\mathbf{p}_c$

The probabilities in Eqs. (19) and (21) can be difficult to compute exactly. Therefore, an approximate solution is considered. Given a unit normal vector $\mathbf{n}$ of the obstacle $\mathcal{O}$ that points into the free workspace (see Fig. 2), 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. (21).

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 along the normal $\mathbf{n}$ into a one-dimensional Gaussian distribution centered at $\mathbf{p}_c$,

$$\mathcal{P}_{\text{collision}} \approx \mathcal{N}(\hat{\mathbf{p}}^\star(\tau), \mathbf{n}^T(P_c(\tau) + Z(\mathbf{x}))\mathbf{n}). \tag{22}$$

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 (19), 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

$$\forall t \in [0, \tau] :: \mathcal{R}(\hat{\mathbf{p}}^\star(t)) \cap (\mathcal{O} \oplus \{\hat{\sigma}\mathbf{n}\}) = \emptyset, \tag{23}$$

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

$$\hat{\sigma} = a\mathbf{n}^T\sqrt{P_c(\tau) + Z(\mathbf{x})}\mathbf{n}, \tag{24}$$

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

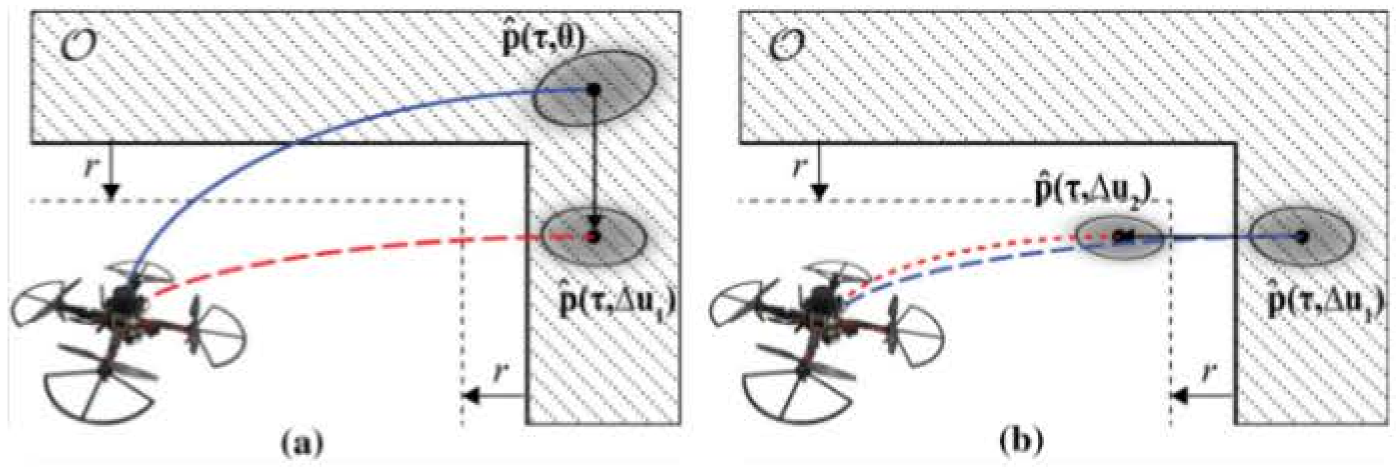

$\mathcal{O}$
$\hat{\mathbf{p}}(\tau,\mathbf{0})$
r
$\hat{\mathbf{p}}(\tau,\Delta\mathbf{u}_1)$
(a)
$\hat{\mathbf{p}}(\tau,\Delta\mathbf{u}_2)$
(b)

Fig. 3 The iterative process of the collision avoidance algorithm: **a** A trajectory, $\hat{\mathbf{p}}(\tau, \mathbf{0})$, is estimated from the original operator's input such that $\Delta\mathbf{u} = \mathbf{0}$. 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, $\mathbf{u} + \Delta\mathbf{u}_1$, is determined to avoid the first detected collision. **b** The trajectory, $\hat{\mathbf{p}}(\tau, \Delta\mathbf{u}_1)$, for the new input is predicted. This also results in a collision and the algorithm computes a new change in input $\Delta\mathbf{u}_2$ with respect halfplane constraints defined by both the initial trajectory $\hat{\mathbf{p}}(\tau, \mathbf{0})$ and $\mathbf{p}(\tau, \hat{\Delta}\mathbf{u}_1)$. 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 (compare block diagram in Fig. 1)

Equations (20) and (21) can now be approximated by the following simplified expression:

$$\mathbf{p}_c = \hat{\mathbf{p}}^\star\left(\operatorname{argmin}_{t\in[0,\tau]}\{\mathcal{R}(\hat{\mathbf{p}}^\star(t)) \cap (\mathcal{O} \oplus \{\hat{\sigma}\mathbf{n}\}) \neq \emptyset\}\right), \tag{25}$$

where if the system has no uncertainty $\hat{\sigma} = 0$, then Eqs. (23) and (25) are equivalent to the deterministic solution in (Israelsen et al. 2014).

Next, given Eqs. (23) and (20), a linear constraint is defined on the position $\hat{\mathbf{p}}(\tau, \Delta\mathbf{u})$ of the robot at time τ (see Eq. (2)),

$$\mathbf{n}^T\hat{\mathbf{p}}(\tau, \Delta\mathbf{u}) > \mathbf{n}^T\mathbf{p}_c. \tag{26}$$

Substituting Eq. (15) from Assumption 3, the constraint on the robot's position in Eq. (26) can be transformed into a constraint on its change in input $\Delta\mathbf{u}$, hence

$$\mathbf{n}^T J(\tau)\Delta\mathbf{u} > \mathbf{n}^T(\mathbf{p}_c - \hat{\mathbf{p}}^\star(\tau)). \tag{27}$$

Equation (13) is approximated using Eq. (27) as

$$\begin{aligned}&\text{minimize: } \Delta\mathbf{u}^T R\Delta\mathbf{u}\\ &\text{subject to: } \mathbf{n}^T J(\tau)\Delta\mathbf{u} > \mathbf{n}^T(\mathbf{p}_c - \hat{\mathbf{p}}^\star(\tau)),\end{aligned} \tag{28}$$

where solving this convex optimization, such as is done by the RVO library in (van den Berg et al. 2011), provides a collision free change in input $\Delta\mathbf{u}$ with the control input given to the robot as $\mathbf{u} + \Delta\mathbf{u}$.

3.2.3 Handling convex corners

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 as described below.

Assume the algorithm has computed a change in control input according to Eq. (13) 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

$$\mathbf{a}_i^T\Delta\mathbf{u} > b_i, \tag{29}$$

where

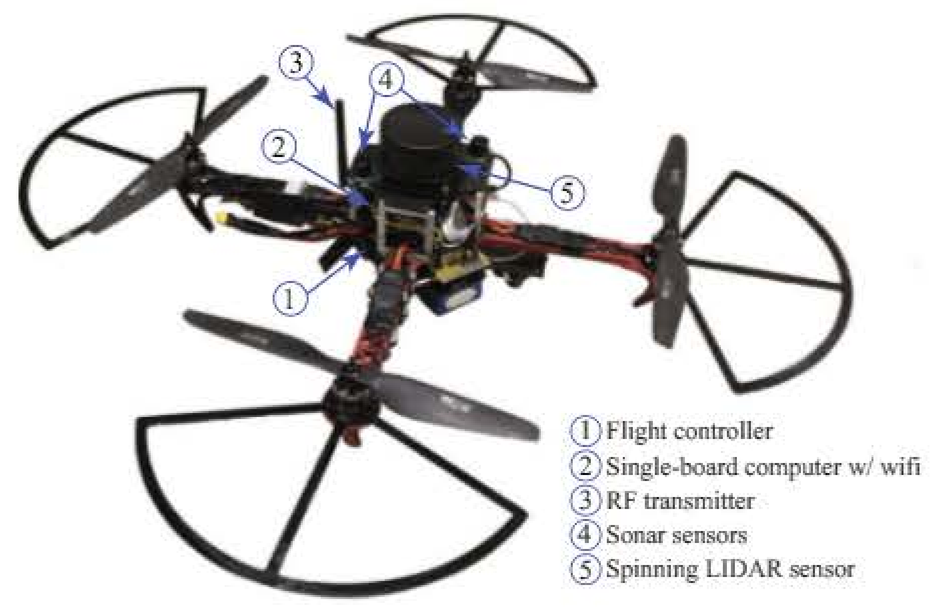
$$\mathbf{a}_i^T = \mathbf{n}_i^T J_i(\tau), \tag{30}$$

$$b_i = \mathbf{n}_i^T(\mathbf{p}_{c,i} - \hat{\mathbf{p}}_i(\tau)). \tag{31}$$

The convex optimization problem in Eq. (28) can now be solved for all iterations i by the following:

$$\begin{aligned}&\text{minimize: } \Delta\mathbf{u}^T R\Delta\mathbf{u}\\ &\text{subject to: } \textstyle\bigcap_{j=1}^{i}\{\mathbf{a}_j^T\Delta\mathbf{u} > b_j\}.\end{aligned} \tag{32}$$

Thus, every ith iteration of the algorithm introduces an additional constraint to the convex optimization problem. After at most d iterations, the control input $\mathbf{u} + \Delta\mathbf{u}$ 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.



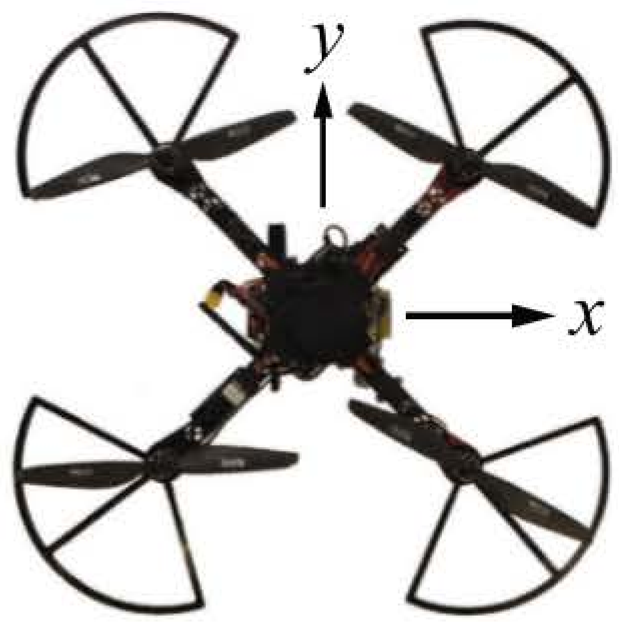


Fig. 4 The custom-designed experimental quadcopter UAV system with on-board obstacle detection hardware shown on the left. The Odroid XU4, running the Robot Operating System (ROS), reads the sensor data and runs the algorithms on-board and provides commands to the Pixhawk autopilot to control the motion of the UAV. On the right shows the *top-down* view of the quadcopter and the x/y coordinate frame in which the 2D spinning LIDAR provides data. The range finders (single-point LIDAR and sonar sensors) are oriented in the z-direction in and out of the page

This iterative approach aligns with the LP-type algorithm in (De Berg et al. 2008). 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.

3.2.4 Incorporating yaw control

For a hover-capable multi-rotor UAV, yaw is a redundant degree-of-freedom that can be held constant. This condition is further emphasized in Sect. 4.1.1. However, when the UAV is equipped with cameras that enable a pilot to survey an area in applications such as a search and rescue, the pilot must be able to rotate the robot. This is particularly important if the pilot is flying through a first person video feed on a forward-facing camera.

The yaw degree-of-freedom is controlled completely by the pilot, meaning the algorithm does not augment this input. As discussed in Sect. 3.2.3, the dimension of the input adjusted by the algorithm is equal to the dimension of the workspace d to ensure the convex optimization problem is feasible. However, the yaw rate of the robot can still be controlled by the user with this algorithm. The algorithm will calculate the feedforward trajectory estimate assuming a constant yaw-rate (from the user) and will calculate the new roll, pitch, and thrust at time $t = 0$ to avoid a collision at time τ. The yaw-rate affects the algorithm's change in roll and pitch through the Jacobian in Eq. (16).

It is pointed out that the maximum yaw-rate and time horizon must be carefully selected. If either value is too large the predicted trajectory of the robot can be poorly predicted by Assumption 3 or can violate Assumption 4, possibly leading to collisions.

4 The experimental UAV system with on-board computation and sensing

In this section, the custom-designed experimental quadcopter UAV system is described, along with the on-board obstacle detection hardware and relevant signal processing algorithms.

4.1 The experimental quadcopter UAV system

The collision avoidance and obstacle detection algorithms are implemented on a custom-designed physical quadrotor helicopter shown in Fig. 4. The UAV has a footprint of 75 cm from rotor-tip to rotor-tip. In Fig. 4, the key components are listed, where the quadcopter uses the Pixhawk commercial autopilot from 3D Robotics for flight control. The 2D spinning LIDAR is the RPLidar 360° LIDAR. The RPLidar has a 1° resolution, providing 360 readings, at a frequency of 10 Hz. The update frequency of the LIDAR is low for the potential speeds of the quadrotor, but experiments were able to demonstrate collision avoidance in the less-than-ideal conditions. A faster LIDAR update could allow for faster motion while still avoiding collisions. The LIDAR-Lite laser rangefinder from PulsedLight serves as the downward-facing laser rangefinder. In addition, the system has two upward-facing sonar sensors (Maxbotix XL-EZ4); however, the experiments performed focus on the results of collision avoidance with respect to walls in the environment using the 2D LIDAR rather than the floor and ceiling.

The collision avoidance algorithm is running on an on-board single-board computer (Odroid XU4) at a fixed frequency of 50 Hz. The computer is equipped with the Ubuntu 14.04 operating system running the Robot Operating System (ROS), version Indigo. The Pixhawk and sensors are connected to the Odroid through a USB interface. The Pixhawk receives the pilot's desired roll, pitch, yaw rate, and throttle commands through a standard 2.4 GHz RC transmitter and passes these values to the Odroid. These inputs are then updated if a collision is predicted and the new values are passed from the Odroid to the Pixhawk to control the motion of the robot.

4.1.1 Robot dynamics

The Pixhawk autopilot contains on-board proportional-integral-derivative (PID) controllers to stabilize the attitude of the robot. The Pixhawk also allows for an input of a throttle command. This throttle command is calculated by a velocity controller based on the pilot's throttle stick command. The closed-loop model is used to calculate the feedforward trajectory estimate provided an estimate of the current state through an on-board Kalman Filter, implemented on the Odroid. The model has a 12-dimensional state that consists of position $\mathbf{p} \in \mathbb{R}^3$, velocity $\mathbf{v} \in \mathbb{R}^3$, Euler RPY angles $\mathbf{r} \in \mathbb{R}^3$, and angular velocity $\mathbf{w} \in \mathbb{R}^3$:

$$\mathbf{x} = [\mathbf{p}^T, \mathbf{v}^T, \mathbf{r}^T, \mathbf{w}]^T \in \mathcal{X}. \tag{33}$$

The 4-dimensional control input consists of the desired roll and pitch angles, $r_x^\star$ and $r_y^\star$ respectively, the vertical velocity $v_z^\star$, and the yaw-rate $w_z^\star$:

$$\mathbf{u} = [r_x^\star, r_y^\star, v_z^\star, w_z^\star]^T \in \mathcal{U}. \tag{34}$$

The equations of motion are given as

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{35}$$

$$\dot{\mathbf{v}} = R\left[0, 0, k_{\text{pv}}(v_z^\star - v_z)\right]^T - \mathbf{g}, \tag{36}$$

$$\dot{\mathbf{r}} = \mathbf{w}, \tag{37}$$

$$\dot{\mathbf{w}} = \begin{bmatrix} k_{\text{px}}(r_x^\star - r_x) - k_{\text{dx}} w_x \\ k_{\text{py}}(r_y^\star - r_y) - k_{\text{dy}} w_y \\ k_{\text{pz}}(w_z^\star - w_z) \end{bmatrix}, \tag{38}$$

where R is the rotation matrix from the robot frame into the world frame, the terms k_{pv}, k_{px}, k_{dx}, k_{py}, k_{dy}, and k_{pz} are gains whose values were determined through system identification of the physical system. These model parameters, given in Table 1, were determined experimentally using a motion capture system to determine the quadcopter's roll and pitch angles in response to step command from the remote controller. These parameters are estimates for the entire closed-loop system from the stick position on the remote control to the measured attitude of the quadrotor. The onboard stabilization controllers were provided by the Pixhawk autopilot system and the provided PID gains were used on the quadrotor system.

Table 1 Quadcopter Dynamic Parameters

Parameter	k_{pv}	k_{px}	k_{dx}	k_{py}	k_{dy}	k_{pz}
Value	10.0	150.0	2.5	150.0	2.5	3.5

4.2 On-board obstacle detection

The algorithm, in general, requires a set of three-dimensional planar faces to perform the collision avoidance. However, to simplify the problem a two-dimensional spinning LIDAR and one-dimensional laser rangefinders are used to detect obstacles. The LIDAR provides a set of 2D data points representing the distance to the nearest object in the x/y plane of the quadcopter body frame (see Fig. 4). These obstacles are assumed to be vertical in the inertial frame such that the 2D data can be used rather than utilizing a 3D point cloud rangefinder, allowing for fast computation with onboard processing without requiring more complex implementations such as GPU processing. This assumption is feasible for situations where the user will not be commanding large vertical velocities at the same time as large roll or pitch commands, possibly creating a trajectory into an unsensed region of the workspace.

4.2.1 LIDAR segmentation

In order to provide a more efficient obstacle representation to the collision avoidance algorithm in this paper, the 2D range data provided by the spinning LIDAR will be segmented through the clustering and split-and-merge algorithm as discussed in (Nguyen et al. 2007). This will reduce the number of planar faces that must be checked for a collision against the predicted trajectory of the robot. The algorithm is presented below in Algorithm 1 and described next.

The clustering process first takes the list of raw range readings from the sensor and separates it into clusters. If two points have a difference in range greater than a predefined magnitude r_{thresh} then they are considered to be two separate sets of data and the list of range data is split between the two example points (Fig. 5a). Next, these subsets from the clustering process are provided to the split-and-merge algorithm. The split-and-merge process considers each set of points and creates a line between the first and last points

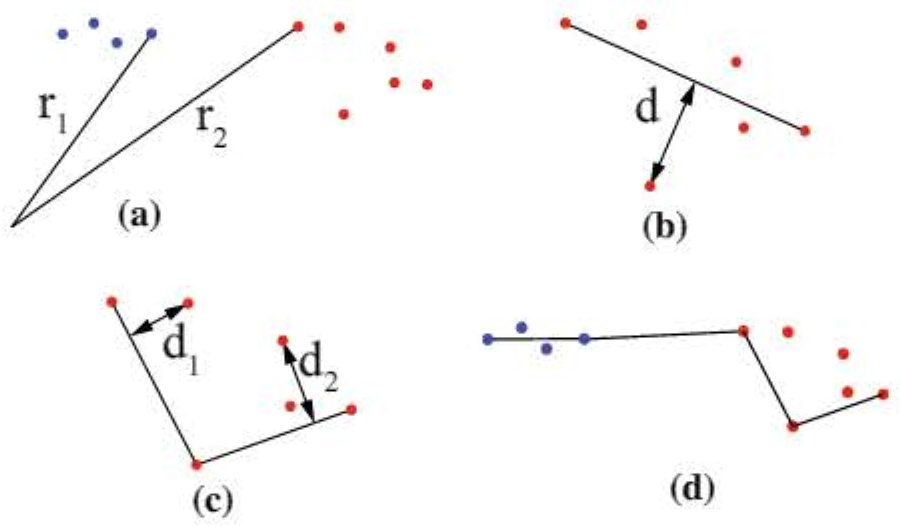


Fig. 5 The graphical representation of the clustering and split-and-merge algorithm for the LIDAR data. **a** First, the data is clustered based on the difference in the range data. If two points have a radial distance greater than a threshold $|r_1 - r_2| > r_{\text{thresh}}$ then they are clustered separately, shown as sets of blue and red points for the two clusters. **b** Shown is the red cluster performing the split-and-merge algorithm. A line is created between the beginning and end of the cluster and the distance d of the point furthest from the line is calculated. If that distance is greater than a threshold $d > d_{\text{thresh}}$, the cluster is split at that point. **c** The results from (**b**) were split to create two separate clusters of points. These two new clusters have their furthest point within the threshold $d_1 < d_{\max}$ and $d_2 < d_{\max}$ as shown. Therefore, the split-and-merge algorithm is complete for that set of clusters. **d** Shown is the result of the clustering and split-and-merge algorithm on the small example data set

in a given cluster. The point between the first and last point in the segment, if one exists, with the maximum distance to the line is selected (Fig. 5b). If that computed maximum distance is greater than some threshold $d_{\max}$, the segment is split into two segments at the point with the maximal distance to the line between the first and last points (Fig. 5c). This process is continued over all subsets until a list of line segments remains where all the data points are within $d_{\max}$ distance from one of the line segments (Fig. 5d). The first and last point in each of the resulting segments are then considered as vertices of the obstacles for the collision avoidance algorithm.

4.2.2 Minkowski difference algorithm

The data that has been processed by the LIDAR segmentation, discussed previously, is used to compute an approximate Minkowski difference which will expand the obstacles by the robot's radius. This new volume, after the Minkowski difference, is provided to the collision avoidance algorithm as the true obstacles to avoid in the environment. A fast Minkowski solution such as those in (Lien 2007; Behar and Lien 2011) can be used if the robot or obstacles are complicated shapes, however, in this application the robot is bounded by the minimum volume sphere. Given the simple geometry of the sphere, the Minkowski difference was implemented approximately and directly to avoid the additional computations of, for example, the reduced convolutions in (Behar and Lien 2011).

Algorithm 1 Clustering and Split-and-Merge

```
$\mathcal{L} \leftarrow s_0, s_1, ..., s_N$ ranges from LIDAR
for all $s_i \in \mathcal{L}$ do
    if $|s_i - s_{i-1}| > r_{\text{thresh}}$ or $|s_i - s_{i+1}| > r_{\text{thresh}}$ then
        Split $\mathcal{L}$ at $s_i$
    end if
end for
for all $\mathcal{L}_j \in \mathcal{L}_0, \mathcal{L}_1, ..., \mathcal{L}_J$ do
    $L \leftarrow$ line between endpoints of $\mathcal{L}_j$
    $d \leftarrow \max(\text{distance}(L, s_k \in \mathcal{L}_j))$
    if $d > d_{\max}$ then
        Split $\mathcal{L}$ at argmax$(\text{distance}(L, s_k \in \mathcal{L}_j))$
    else
        Segment $\mathcal{L}_j$ is complete
    end if
end for
```

4.2.3 Practical considerations

The algorithm previously presented is developed to provide collision-free motion including uncertainty in the forward prediction of the trajectory as well as in the obstacles' sensed positions. In (Bareiss et al. 2015), the algorithm was studied for varying amounts of uncertainty and confidence bounds. Provided the uncertainty covariance matrices, the experimental application in this paper could follow that approach directly, however, during preliminary experiments the performance of the algorithm was found to be dominated by other factors discussed next and these covariances were not implemented directly.

When the quadcopter is some small distance from the wall and uncertainty causes it to move closer to the wall than the *offset* distance value $\hat{\sigma}$ in Eq. (24), the algorithm will provide a small desired roll and pitch angles. The Pixhawk autopilot was found to be unable to respond to these small inputs as they are within the signal-to-noise ratio from the lower-quality sensors hardware. This could lead the quadcopter to not physically respond to the algorithm's desired output. Secondly, this application on a multi-rotor vehicle in constrained indoor environments is also subject to aerodynamic disturbances which can be large and difficult to model as shown in current research (Hooi et al. 2015; Yeo et al. 2016). These disturbances will also likely violate the assumptions in (Bareiss et al. 2015) that the noise can be modeled as normal distribution.

The lack of control authority for low-angle desired roll and pitch as well as aerodynamic disturbances were found to dominate the performance of the algorithm, leading to collisions. Therefore, through preliminary experiments a constant *offset* distance value of $\hat{\sigma} = 1.2$ m was empirically selected. Even in situations with the combined effects of the state estimate error, lack of control authority, and aerodynamic disturbances, this value of $\hat{\sigma}$ was observed to be able to keep the quadcopter free of collisions. Provided a perfect model, the robot would never achieve a distance to an obstacle less than the offset $\hat{\sigma}$. However, in the presence of the mentioned uncertainties, the robot can potentially overshoot this value and move closer to obstacles than value, but will still avoid collisions.

5 Experimental results and discussion

The model-based collision avoidance algorithm was implemented on the experimental quadcopter system to demonstrate the performance of the algorithm. In particular, four cases were studied. In the first case the pilot commanded the aerial robot to fly straight at a wall. In the second case, the pilot flew the robot into a corner. In the third case, the pilot flew the robot through a zone with internal obstacles. Finally, in the fourth experiment the robot was flown through a hallway with an "S" turn. In each experiment, the data from the spinning LIDAR was collected as well as the output of the split-and-merge segmentation and the approximate Minkowski difference. The initial desired trajectory from the pilot's input is also recorded along with the final, collision free trajectory the algorithm calculates. In all four cases, the robot executed the algorithm and performed as expected, and the measured responses also agreed with simulations and expected behaviors. The results are described next.

In first case, the quadcopter was flown straight at a wall and the results are shown in Fig. 6. The series of images shown in the first two columns in Fig. 6 present a sequence of side- and top-view images extracted from recorded video during the experiment. The results on the far right-hand column shows the recorded raw LIDAR points, segmented points, Minkowski points, and the initial and final trajectories of the robot at the corresponding time steps. The red arrows show the desired trajectory given the user's input while the green arrows show the resultant trajectory from the algorithm. Also, it can be seen in the sensor data in the right column that there are times when the LIDAR returns a maximum range reading when it should be located on the obstacle. This is from errors in the LIDAR that can be filtered with post-processing, however, in this paper these "blips" in the scan are accounted for in the Minkowski difference and do not need to be filtered from the LIDAR data initially. As can be seen, initially at $t = 2\,\mathrm{s}$ the robot is commanded to fly into the wall on the left, indicated by the red arrow. Since the wall was sufficiently far from the robot, the algorithm did not alter the pilot's control input. But as the relative distance between the wall and robot began to shrink ($t = 4, 6, 8\,\mathrm{s}$) and a possible collision was detected by the on-board LIDAR sensor, the algorithm began to adjust the control input such that it forced the robot to slow down, and eventually come to a stop in front of the wall in the limiting case, irrespective of the pilot continuing to command the robot towards the wall. The motion away from the wall rather than completely stopping is a result of uncertainty in the motion model causing the robot to pass the safety bound ($t = 6, 8\,\mathrm{s}$) of the algorithm and have to reverse ($t = 10\,\mathrm{s}$), overshooting the desired position. A more accurate state estimate through additional sensors would reduce the magnitude of this overshoot. Based on the results in this first case, the robot can automatically detect an obstacle (such as the wall) along its trajectory and the algorithm altered the control input to avoid a collision.

In the second case, the quadcopter was flown at a corner and the experimental results are shown in Fig. 7. As shown, the robot was first flown towards a wall on its right and then it strafed along that wall into the corner ($t = 2, 4\,\mathrm{s}$), followed by strafing along the wall in front of the robot ($t = 6, 8, 10\,\mathrm{s}$) as shown in the sequence of images in the first column in Fig. 7. The resulting behavior is collision-free motion with respect to both of the walls. Again, it can be seen in the physical sensor data in the right-hand column in Fig. 7 that there are times when the LIDAR returns a maximum range reading when it should be located on the obstacle, but these errors are accounted for in the Minkowski difference. Similar to the results from Case 1, the robot can automatically detect surrounding obstacles in the x/y plane (such as two walls that form a corner) along its trajectory and modify the control input to avoid a collision.

In the third case, the robot was tested to determine it's ability to handle internal obstacles (such as office file cabinets). As shown in Fig. 9, the quadcopter was flown at a narrow obstacle that is within the environment rather than only using the exterior walls. The quadcopter flies towards the cabinet obstacle ($t = 0, 1\,\mathrm{s}$) until it then strafes to the right along the front face of the cabinet ($t = 2, 3\,\mathrm{s}$). After passing the cabinet, the quadcopter is able to move towards the wall and come to a stop ($t = 4 - 9\,\mathrm{s}$).

Finally, in the fourth case, both simulations and experiments were performed to study the performance of the algorithm in a more natural environment such as flying through an "S"-shaped basement hallway. Simulation of the hallway scenario was created within a deterministic simulation environment (V-REP, Coppelia Robotics) to be compared to the real-world experimental results. The simulation used the robot model presented above in Sect. 4.1.1 and the collision avoidance algorithm presented herein. The simulation and experimental results are shown in Fig. 8a, b, respectively. The results also include the simulated and measured raw LIDAR points, segmented points, Minkowski points, and the initial and final trajectories of the robot at the corresponding time steps. As shown in both the simulation and experimental results, the quadcopter was flown from a straight hallway towards a wall ($t = 0, 2, 4, 6\,\mathrm{s}$) where it strafes to the left ($t = 8, 10, 12, 13\,\mathrm{s}$), then the space opened up into a straight hallway and the robot continues to fly down the straight hallway ($t = 14, 16\,\mathrm{s}$). Both the simulated trajectories and LIDAR scan data showed good agreement with

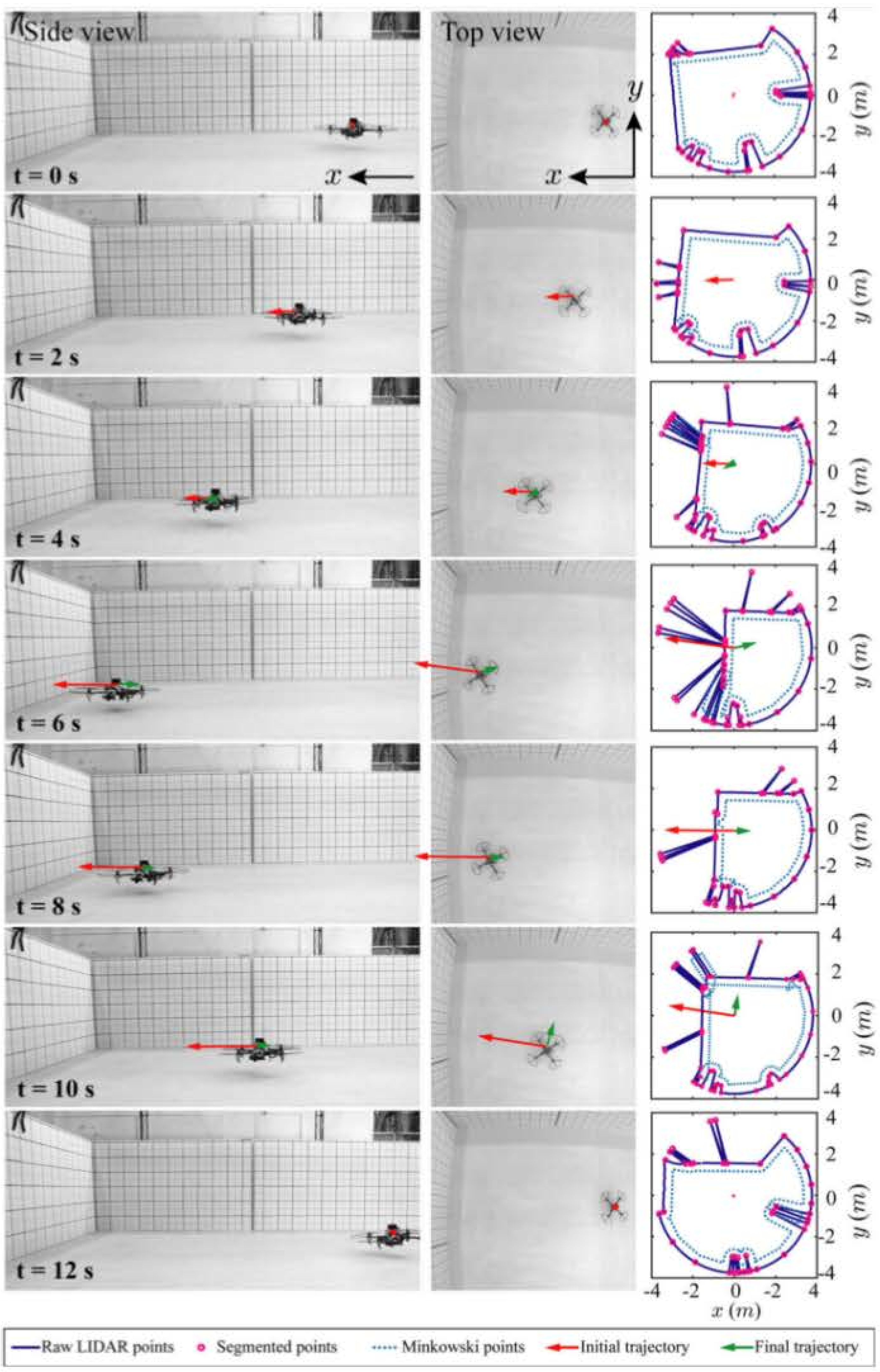


Fig. 6 Case 1: Experimental results where the aerial robot was flown directly at a wall in front of it. A sequence of time steps is shown along with the data from the sensors and the resultant desired trajectories. The left column shows the side view were the quadcopter moves towards the wall and then back away from it. The motion away from the wall rather than completely stopping is a result of uncertainty in the motion model causing the robot to pass the safety bound ($t = 6, 8$ s) of the algorithm and have to reverse ($t = 10$ s), overshooting the desired position. A more accurate state estimate through additional sensors would reduce the magnitude of this overshoot

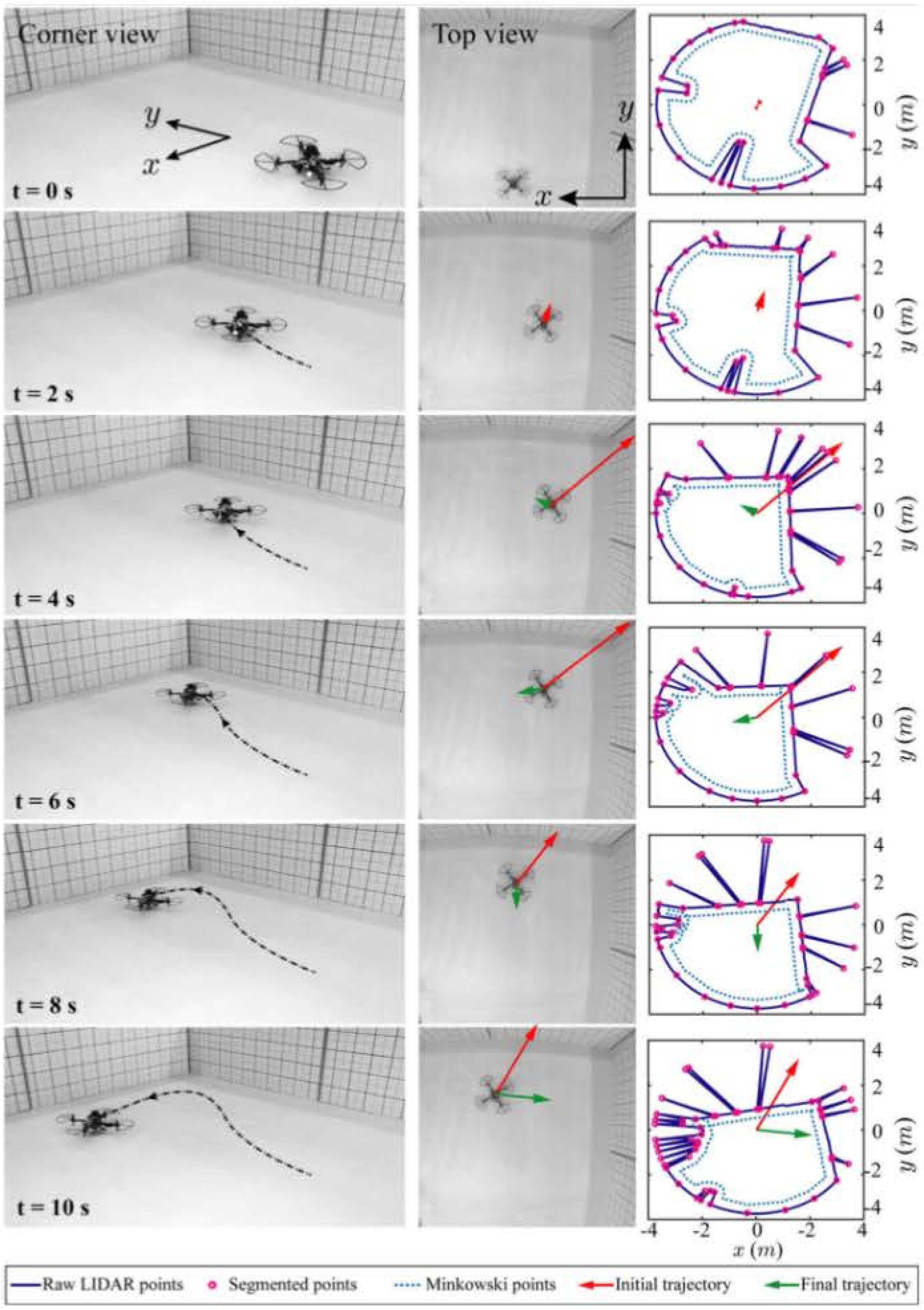


Fig. 7 Case 2: Experimental results where the aerial robot was flown into a corner. The quadcopter is first flown towards the wall on the right where it then strafes along it temporarily ($t = 2, 4$ s) before then moving along the wall in front of it ($t = 6, 8, 10$ s), resulting in collision-free motion with respect to both of the walls

the measured experimental results shown in Fig. 8b. Thus, the results in this case shows that the robot can automatically detect and react to obstacles along its trajectory and the results also demonstrate application of the collision avoidance algorithm in a natural environment (Fig. 7).

6 Conclusions and future work

In this paper, a feedforward-based automatic collision avoidance algorithm was presented and implemented on an experimental quadcopter with on-board sensing and computation.

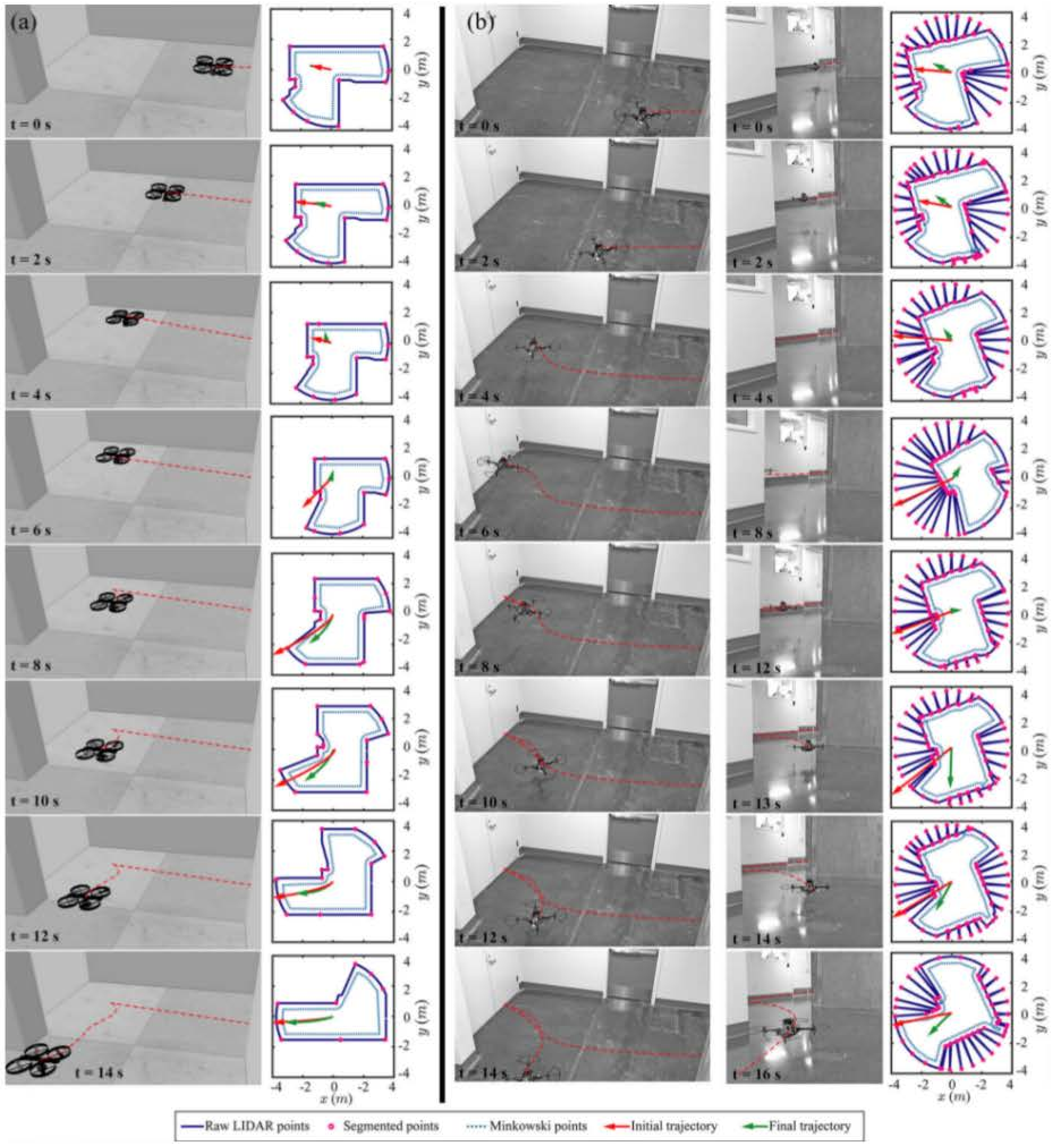


Fig. 8 Case 4: **a** Simulations and **b** experiments of the robot flying through an "S"-shaped basement hallway. The quadcopter was flown from a straight hallway towards a wall ($t = 0, 2, 4, 6$ s) where it strafes to the left ($t = 8, 10, 12, 13$ s), then the space opened up into a straight hallway and the robot continues to fly down the straight hallway ($t = 14, 16$ s)

From a pilot's input, the algorithm predicts the trajectory given its current state that the robot will follow if the input remains constant over some time horizon. If there is a probability for a collision along the trajectory greater than some predetermined bound, considering uncertainty in the robot's motion model and sensing accuracy, then the algorithm modifies the pilot's input for a new, collision-free input. A 2D spinning LIDAR was used to obtain the planar distances to

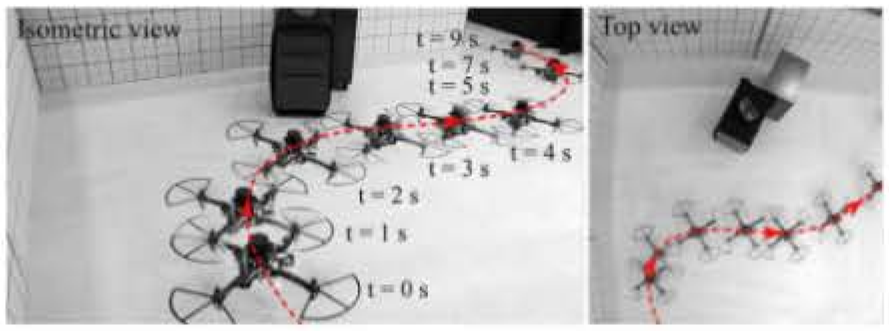


Fig. 9 Case 3: In this experiment, the quadcopter is flown at a narrow obstacle that is within the environment rather than only using the horizontal walls. The quadcopter flies towards the cabinet ($t = 0$, 1s) until it then strafes to the right along the front face of the cabinet ($t = 2$, 3 s). After passing the cabinet, the quadcopter is able to move towards the wall and come to a stop ($t = 4 - 9$ s)

objects in the environment. This range data was processed using a clustering and split-and-merge algorithm to reduce the number of planar faces to be considered in the collision avoidance algorithm. An approximate Minkowski difference of these planar faces was then used to avoid collisions in real-time operation. The implementation was tested in a variety of environments to demonstrate its performance. The quadcopter was shown to avoid collisions even when the pilot was intentionally controlling it towards a collision with obstacles in the environment.

In the future, this algorithm could be improved by including an adaptive model for the feedforward trajectory estimate as well as including additional sensing to provide a more accurate prediction of collisions. A more accurate collision prediction through improved models of the aerodynamics as well as more state estimates for the on-board attitude controllers would allow for the safety buffer to be decreased which would cause the robot to fly closer to obstacles and with higher speeds.

Acknowledgements 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

Abdilla, A., Richards, A., & Burrow, S. (2015). Power and endurance modelling of battery-powered rotorcraft. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 675– 680).

Achtelik, M. W., Lynen, S., Weiss, S., Chli, M., & Siegwart, R. (2014). Motion- and uncertainty-aware path planning for micro aerial vehicles. *Journal of Field Robotics*, *31*(4), 676–698.

Adams, M., Wijesoma, W. S., & Shacklock, A. (2007). Autonomous navigation: Achievements in complex environments. *IEEE Instrumentation and Measurement*, *10*(3), 15–21.

Agrawal, P., Ratnoo, A., & Ghose, D. (2015). Vision based obstacle detection and avoidance for UAVs using image segmentation. In *AIAA guidance, navigation, and control conference* (pp. 848–857).

Astilla, O., Guerrero, J., Mendoz, R., Teriz, P., & Roxas, M .(2015). Obstacle avoidance of hybrid mobile-quadrotor vehicle with range sensors using fuzzy logic control. In *International conference on humanoid, nanotechnology, information technology, communication and control, environment and management* (pp. 1–8).

Bareiss, D., van den Berg J, & Leang, K. K. (2015). Stochastic automatic collision avoidance for tele-operated unmanned aerial vehicles. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 4818–4825).

Barrientos, A., Colorado, J., Cerro, J. D., Martinez, A., Rossi, C., Sanz, D., et al. (2011). Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, *28*(5), 667–689.

Behar, E., & Lien, J. M. (2011). Fast and robust 2d minkowski sum using reduced convolution. In: *IEEE/RSJ international conference on intelligent robots and systems* (pp. 1573–1578).

Bernini, N., Bertozzi, M., Castangia, L., Patander, M., & Sabbatelli, M. (2014). Real-time obstacle detection using stereo vision for autonomous ground vehicles: a survey. In *IEEE Internationl conference on intelligent Transportation Systmes* (pp. 873–878).

Borenstein, J., & Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, *7*(3), 278–288.

Brand, C., Schuster, M. J., Hirschmuller, H., & Suppa, M. (2014). Stereo-vision based obstacle mapping for indoor/outdoor SLAM. In: *IEEE/RSJ International conference on intelligent robots and systems* (pp. 1846–1853).

Brown, T., Doshi, S., Jadhav, S., & Himmelstain, J. (2004). Test bed for a wireless network on small UAVs. In: *AIAA 3rd "Unmanned Unlimited" technical conference, workshop, and exhibit* (pp. 20–23).

Chryssanthacopoulos, J. P., & Kochenderfer, M. J. (2011). Accounting for state uncertainty in collision avoidance. *Journal of Guidance, Control, and Dynamics*, *34*(4), 951–960.

Cole, D. T., Sukkarieh, S., & Göktoğan, A. H. (2006). System development and demonstration of a uav control architecture for information gathering missions. *Journal of Field Robotics*, *23*(6–7), 417–440.

Cook, Z., Zhao, L., Lee, J., & Yim, W. (2015). Unmanned aerial system for first responders. In: *12th international conference on ubiquitous robots and ambient intelligence (URAI)* (pp. 306–310).

D'Attanasio, S., Tonet, O., Megali, G., Carrozza, M. C., & Dario, P. (2000). A semi-automatic handheld mechatronic endoscope with collision-avoidance capabilities. In: *IEEE international conference on robotics and automation* (pp. 1586–1591).

De Berg, M., Cheong, O., van Kreveld, M., & Overmars, M. (2008). *Computational Geometry: Algorithms and Applications*. Berlin: Springer.

Fiorini, P., & Shiller, Z. (1998). Motion planning in dynamic environmnts using velocity obstacles. *International Journal of Robotics Research*, *17*(7), 760–772.

Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, *4*(1), 23–33.

Gatti, M., Giulietti, F., & Turci, M. (2015). Maximum endurance for battery-powered rotary-wing aircraft. *Aerospace Science and Technology*, *45*, 174–179.

Goodrich, M. A., Morse, B. S., Gerhardt, D., Cooper, J. L., Quigley, M., Adams, J. A., et al. (2008). Supporting wilderness search and rescue using a camera equipped mini UAV. *Journal of Field Robotics*, *25*(1–2), 89–110.

Han, J., Xu, Y., Di, L., & Chen, Y. (2013). Low-cost multi-UAV technologies for contour mapping of nuclear radiation field. *Journal of Intelligent & Robotic Systems*, *70*(1), 401–410.

Hausamann, D., Zirnig, W., Schreier, G., & Strobl, P. (2005). Monitoring of gas pipelines a civil UAV application. *Aircraft Engineering and Aerospace Technology*, *77*(5), 352–360.

Hooi, C. G., Lagor, F. D., & Paley, D. A. (2015). Flow sensing, estimation and control for rotorcraft in ground effect. In *Proceedings of the IEEE aerospace conference* (pp. 1 – 8).

Huh, K., Park, J., Hwang, J., & Hong, D. (2008). A stereo vision-based obstacle detection system in vehicles. *Optics and Lasers in engineering*, *26*(2), 168–178.

Israelsen, J., Beall, M., Bareiss, D., Stuart, D., Keeney, E., & van den Berg, J. (2014). Automatic collision avoidance for manually tele-operated unmanned aerial vehicles. In: *IEEE international conference on robotics and automation* (pp. 6638–6643).

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, *5*(1), 90–98.

Kumar, M., Cohen, K., & Homchaudhuri, B. (2011). Cooperative control of multiple uninhabited aerial vehicles for monitoring and fighting wildfires. *Journal of Aerospace Computing, Information, and Communication*, *8*(1), 1–16.

Landis, G. A. (2004). Robots and humans: Synergy in planetary exploration. *Acta astronautica*, *55*(12), 985–990.

Lien, J. M. (2007). Point-based minkowski sum boundary. In *15th Pacific conference on computer graphics and applications* (pp. 261–270).

Lin, P. S., Hagen, L., Valavanis, K., & Zhou, H. (2005). Vision of unmanned aerial vehicle (UAV) based traffic management for incidents and emergencies. In *12th World congress on intelligent transport systems* (pp. 1–12).

Li, S., & Tao, G. (2009). Feedback based adaptive compensation of control system sensor uncertainties. *Automatica*, *45*(2), 393–404.

Mahony, R., Kumar, V., & Corke, P. (2012). Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics & Automation Magazine*, *19*(3), 20–32.

Maier, D., Hornung, A., & Bennewitz, M. (2012). Real-time navigation in 3d environments based on depth camera data. In *International conference on humanoid robots* (pp. 692–697).

Matthies, L., Brockers, R., Kuwata, Y., & Weiss, S. (2014). Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In *IEEE international conference on robotics and automation* (pp. 3242–3249).

Mejias, L., McNamara, S., Lai, J., & Ford, J. (2010). Vision-based detection and tracking of aerial targets for UAV collision avoidance. In: *IEEE/RSJ International conference on intelligent robots and systems* (pp. 87–92).

Mendes, J., & Ventura, R. (2013). Assisted teleoperation of quadcopters using obstacle avoidance. *Journal of Automation, Mobile Robotics, & Intelligent Systems*, *7*(1), 54–58.

Mohammed, F., Idries, A., Mohamed, N., Al-Jaroodi, J., & Jawhar, I. (2014). UAVs for smart cities: Opportunities and challenges. In *International conference on unmanned aircraft systems (ICUAS)* (pp. 267 – 273).

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). Fast-SLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI-02 proceedings* (pp. 593–598).

Muller, J., & Sukhatme, G. S. (2014). Risk-aware trajectory generation with application to safe quadrotor landing. In: *IEEE/RSJ international conference on intelligent robots and systems* (pp. 3642–3648).

Neri, M., Campi, A., Suffritti, R., Grimaccia, F., Sinogas, P., & Guye, O et al. (2011). SkyMedia-UAV-based capturing of HD/3D content with WSN augmentation for immersive media experiences. In: *IEEE international conference on multimedia and expo (ICME)* (pp. 1–6). doi:10.1109/ICME.2011.6012133

Nex, F., & Remondino, F. (2013). UAV for 3D mapping applications: A review. *Applied Geomatics*, *6*(1), 1–15.

Nguyen, V., Gachter, S., Martinelli, A., Tomatis, N., & Siegwart, R. (2007). A comparison of line extraction algorithms using 2d range data for indoor mobile robotics. *Autonomous Robots*, *23*(2), 97–111.

Niewenhuisen, M., & Behnke, S. (2015). 3d planning and trajectory optimization for real-time generation of smooth MAV trajectories. In: *European conference on mobile robots* (pp. 1–7).

Patil, S., van den Berg, J., & Alterovitz, R. (2012). Estimating probability of collision for safe planning under gaussian motion and sensing uncertainty. In: *IEEE international conference on robotics and automation* (pp. 3238–3244).

Rehmtullah, F., & Kelly, J. (2015). Vision-based collision avoidance for personal aerial vehicles using dynamic potential fields. In: *12th conference on computer and robot vision* (pp. 297–304).

Rodriguez-Seda, E. J., Stipanovic, D. M., & Spong, M. W. (2011). Collision avoidance with sensing uncertainties. In *American control conference* (pp. 3363–3368).

Saha, S., Natraj, A., & Waharte, S. (2014). A real-time monocular vision-based frontal obstacle detection and avoidance for low cost UAVs in GPS denied environment. In: *IEEE International conference on aerospace electronics and remote sensing technology* (pp. 189–195).

Stegagno, P., Basile, M., Bulthoff, H. H., & Franchi, A. (2014). A semi-autonomous UAV platform for indoor remote operation with visual and haptic feedback. In *IEEE International conference on robotics and automation* (pp. 3862–3869).

Tomic, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., et al. (2012). Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. *IEEE Robotics and Automation Magazine*, *19*(3), 46–56.

Trammell, H. S., Perry, A. R., Kumar, S., Czipott, P. V., Whitecotton, B. R., & McManus, T. J., et al. (2005). Using unmanned aerial vehicle-borne magnetic sensors to detect and locate improvised explosive devices and unexploded ordnance. In *Proceedings of the SPIE sensors, and command, control, communications, and intelligence (C3I) technologies for homeland security and homeland defense IV, vol. 5778.*

Valavanis, K. P., & Vachtsevanos, G. J. (2014). *UAV Sense, Detect and Avoid: Introduction*. Netherlands: Springer.

van den Berg, J., Guy, S. J., Lin, M., & Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Proceedings of the international symposium on robotics research (pp. 3–19).*

Waharte, S., & Trigoni, N. (2010). Supporting search and rescue operations with UAVs. In *IEEE International Conference on Emerging Security Technologies (EST).*

Wang, T., Bu, L., & Huang, Z. (2015). A new method for obstacle detection based on kinect depth image. In: *Chinese automation congress*(pp. 537–541).

Wang, W. P. (1990). Three-dimensional collision avoidance in production automation. *Computers in Industry*, *15*(3), 169–174.

Yeo, D. W., Sydney, N., & Paley, D. A. (2016). Onboard flow sensing for rotary-wing uav pitch control in wind. In: *AIAA guidance, navigation, and control conference* (pp. 1386–1396).

Yoshimoto, H., Jo, K., & Hori, K. (2009). Toward entertainment blimps for everyone by everyone. In *Proceedings of the seventh ACM conference on creativity and cognition* (pp. 445–446).

Daman Bareiss received his B.S. from Oklahoma State University in 2011, M.Sc. and Ph.D. degrees in robotics from the University of Utah, 2014 and 2016, respectively. His research focuses on collision avoidance methods for single- and multi-robot systems. He was the recipient of the NSF IGERT Traineeship for Biocentric Robots, the ARCS Foundation Fellowship, and the University of Utah Graduate Research Fellowship. He is currently a software engineer at Omron-Adept focusing on mobile robot collision avoidance.

Joseph R. Bourne received his B.S. degree from the University of Utah in 2015. He is currently a Ph.D. student in the Department of Mechanical Engineering working in the DARC (Design, Automation, Robotics & Control) Lab, University of Utah Robotics Center. His research focuses on control and motion planning for mobile (aerial and ground) robots with application in autonomous environmental monitoring, search and rescue, and first response.

Kam K. Leang received the B.S. and M.S. degrees in Mechanical Engineering from the University of Utah, Salt Lake City, Utah, in December 1997 and 1999, respectively, and the Ph.D. degree from the University of Washington, Seattle, Washington, in December 2004. He is an Associate Professor in the Mechanical Engineering Department at the University of Utah, where he joined in July 2014. He is the Director of the DARC (Design, Automation, Robotics & Control) Lab and a member of the University of Utah Robotics Center. Between 2008 and 2014, he was at the University of Nevada, Reno. He currently serves as an Associate Editor for IEEE Control Systems Magazine, Mechatronics (Elsevier), the International Journal of Intelligent Robotics and Applications (IJIRA), and Frontiers in Mechanical Engineering (Nature Publishing). He has been involved with conference organization and editorialship activities, including the American Control Conference (ACC), IEEE International Conference on Robotics and Automation (ICRA), and IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM). His research interests include modeling and precision control of electroactive (smart) material actuators (piezoelectrics and electroactive polymers), nanopositioning and scanning probe microscopy, and design, motion planning, and control of robotic systems. He is a member of the ASME and IEEE.