2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Automatic Collision Avoidance for Manually Tele-operated Unmanned Aerial Vehicles

Jason Israelsen Matt Beall Daman Bareiss Daniel Stuart Eric Keeney Jur van den Berg

***Abstract*— In this paper we present an approach that aids the human operator of unmanned aerial vehicles by automatically performing collision avoidance with obstacles in the environment so that the operator can focus on the global direction of motion of the vehicle. As opposed to systems that override operator control as a last resort in order to avoid collisions (such as those found in modern automobiles), our approach is designed such that the operator can *rely* on the automatic collision avoidance, enabling intuitive and safe operator control of vehicles that may otherwise be difficult to control. Our approach continually extrapolates the future flight path of the vehicle given the current operator control input. If an imminent collision is predicted our algorithm will override the operator's control input with the nearest control input that will actually let the vehicle avoid collisions with obstacles. This ensures safe flight while simultaneously maintaining the intent of the human operator as closely as possible. We successfully implemented our approach on a physical quadrotor system in a laboratory environment. In all experiments the human operator failed to crash the vehicle into floors, walls, ceilings, or obstacles, even when deliberately attempting to do so.**

I. Introduction

The use of tele-operated unmanned aerial vehicles (UAVs) in applications such as search and rescue, policing, cinematography, monitoring, entertainment, and inspection of inaccessible or hazardous locations [1], [2], [3], [4] has increased dramatically in recent years, as they have allowed for accessing hard to reach locations both indoor and outdoor. Consider a scenario where a human operator maneuvers a UAV equipped with cameras through a building at risk of a collapse in search of survivors after an earthquake or fire, or through an industrial building where toxic, nuclear, or otherwise hazardous material has been released. In these cases, the operator must make high-level decisions about where to fly the vehicle in potentially unknown indoor environments, and simultaneously ensure that the vehicle does not crash into obstacles, walls, floors and ceilings. UAVs can be difficult to fly even for trained operators, particularly in indoor GPS-denied environments where the operator must navigate with live camera-feed from the vehicle.

To aid the human operator in such tasks, we present an approach that lets the vehicle automatically perform collision avoidance, such that the operator can focus all of their attention on the global decision making. Whereas collision avoidance systems such as those that can be found in modern automobiles warn the driver or even override operator control as a last resort [5], [6], [7], [8], our approach is designed specifically so that the operator can *rely* on the automatic collision avoidance. Our system ensures that collisions are avoided. It also maintains the objective of the operator by continually selecting a control input that is as close as possible to the operator's control input, resulting in an intuitive control interface. This is not unlike the concept of *virtual fixtures* [9] that are commonly used in surgical robotics [10], [11], [12], [13], [14]; our approach similarly allows the operator to use the obstacles for navigation and achieve smooth obstacle-compliant flight (see Fig. 1).

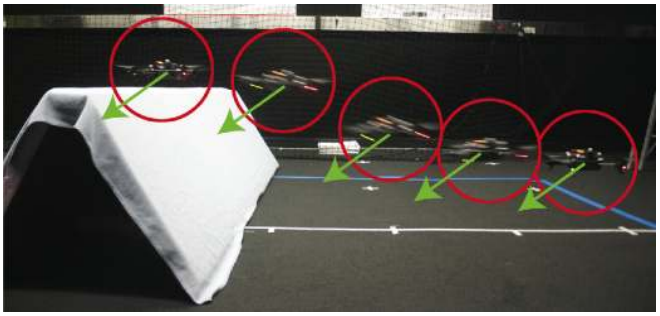

Fig. 1. A time-lapsed picture showing the working of our automatic collision avoidance algorithm for a manually tele-operated quadrotor helicopter. The vehicle was given an operator input to fly into the floor and into the obstacle (green arrows). The automatic collision avoidance system follows the operator inputs as closely as possible while ensuring that collisions are avoided. We refer the reader to the video accompanying this paper and to **http://arl.cs.utah.edu/research/aca** for videos of our experiments.

Specifically, our method continually estimates the future trajectory of the vehicle given its dynamics, its current state, and the current operator's control input. It checks whether a collision will occur with any obstacle within a preset time horizon. If a collision is imminent, our method selects a new control input such that the change to operator's input is minimized while ensuring that collisions are avoided. We note that if no collision is imminent, our method does not change the operator's control input. Our approach is conceptually designed to fit the paradigm in which the vehicle detects the relative location of nearby obstacles in real-time using on-board sensors [15], [16] (our approach does not require absolute positioning). However, in this paper we focus on collision avoidance and assume that the robot knows its local environment and its relative position within.

We implemented our algorithm on a physical quadrotor helicopter within a laboratory environment, highlighting our approach's ability to handle systems with non-linear dynamics that are difficult to operate manually. In our implementation,

Jason Israelsen, Matt Beall, Daman Bareiss, Dan Stuart, and Eric Keeney are with the Department of Mechanical Engineering at the University of Utah. Email: {jason.israelsen, matt.beall, daman.bareiss, dan.stuart, eric.keeney}@utah.edu.

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

positioning of the vehicle is provided by a motion capture system, and the obstacle geometry is preprogrammed. We performed experiments for various scenarios in an indoor environment with walls, floors, ceilings, and obstacles. In all cases the human operator failed to crash the vehicle even when deliberately attempting to do so.

The remainder of this paper is structured as follows. We discuss related work in Section II. The problem we address is formally defined in Section III. Our approach along with implementation details is presented in Section IV. Simulation and experimental results are presented in Section V. We conclude in Section VI.

II. Related Work

The problem of collision avoidance has a long history in robotics. Methods based on artificial potential fields [17], [18], [19], dynamic windows [20], [21], velocity obstacles [22], [23], [24], inevitable collision states [25], and others have been developed over the past decades. Approaches geared specifically to (teams of) unmanned aerial vehicles include [1], [2], [26], [27], [28], [29]. These methods mostly focus on cases where the robot navigates fully autonomously.

Collision avoidance as a means of assisting a human operator has been studied and commercially implemented in automobiles, in which case the system warns the operator or overrides operator control when a collision or unsafe situation is imminent [5], [6], [7], [8]. These systems are typically implemented as last resort safety mechanisms.

Systems designed such that the operator can rely on collision avoidance have mainly been studied in the context of tele-operated (surgical) manipulator robots [10], [11], [12], [13], [14]. These approaches use the concept of *virtual fixtures* [9], which are virtual boundaries in the workspace that aid the operator in moving the robot's end-effector relative to obstacle geometry. The end-effector is prevented from penetrating the fixtures by projecting the range of the robot's Jacobian matrix onto the subspace parallel to the fixtures [10], so that any control input may only result in motion parallel to the fixture. In many cases this is supplemented with haptic feedback to the operator [11], [13]. The goal of our approach is to achieve similar results for tele-operated unmanned aerial vehicles, whose highly inert dynamics is a critical complicating factor that must specifically be accounted for.

Several operator-assisting collision avoidance methods have been developed specifically for UAVs. The method of [30] provides information about the environment to the operator through force feedback. This force feedback is relative to the time to impact (determined from vehicle's current velocity and distance to obstacle), and helps drive the operator from obstacles. Similarly, in [3], the operator input is overridden proportional to an approximate TTC (time to collision), based on the vehicle's current velocity and distance to the obstacle. The TTC is classified as a threat level, and each threat level determines a response (e.g. no action, slow, stop, evasive maneuver). In this paper we take a slightly more principled approach. To aid the operator we specifically take into account the actual dynamics, state, and operator control input to determine collisions in the future from potentially non-linear trajectories, and use this information to minimize the deviation from the operator input such that collisions will not occur.

III. Problem Definition

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

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{1}$$

where t is the time, $\mathbf{x}(t)$ is the state at time t, and $\mathbf{u}(t)$ is the operator input at time t. Then, given the current state $\mathbf{x}_0 = \mathbf{x}(0)$ and a constant operator input $\mathbf{u}$, the state of the robot for $t \geq 0$ is defined by:

$$\mathbf{x}(t) = \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \tag{2}$$

where $\mathbf{g} \in \mathcal{X} \times \mathcal{U} \times \mathbb{R} \to \mathcal{X}$ represents the solution to differential equation (1).

Let $\mathbb{R}^d$, where typically $d = 3$, be the workspace in which the robot maneuvers, and let $\mathcal{O} \subset \mathbb{R}^d$ define the subset of the workspace occupied by obstacles and, in order to be conservative and compatible with the paradigm of on-board sensing, those regions of the workspace that are occluded by the obstacles as seen from the current state of the robot (see Fig. 2). Let $\mathcal{R}(\mathbf{x}) \subset \mathbb{R}^d$ denote the subset of the workspace occupied by the robot when it is in state $\mathbf{x} \in \mathcal{X}$.

The problem we discuss in this paper is now defined as finding a minimal change $\Delta\mathbf{u} \in \mathcal{U}$ to the operator's control input $\mathbf{u} \in \mathcal{U}$ given the current state $\mathbf{x}$ of the robot, such that for all times up to a preset time horizon $\tau \in \mathbb{R}$ the robot does not collide with obstacles:

$$\begin{aligned} \text{minimize:} \quad & \Delta\mathbf{u}^T R \Delta\mathbf{u} \\ \text{subject to:} \quad & \forall t \in [0, \tau] :: \mathcal{R}(\mathbf{g}(\mathbf{x}, \mathbf{u} + \Delta\mathbf{u}, t)) \cap \mathcal{O} = \emptyset, \end{aligned} \tag{3}$$

where $R \in \mathbb{R}^{m \times m}$ is a positive-definite weight matrix.

IV. Approach

In this section we outline our approach. We first discuss simplifying assumptions to make the problem tractable, and then describe our (approximate) solution to the problem.

A. Simplifying Assumptions

The optimization problem as defined by Eq. (3) is highly non-linear and non-convex. Given the real-time constraints of our system, we make a number of key simplifying but reasonable assumptions to make the problem solvable.

Firstly, we assume that the position of the robot, $\mathbf{p} \in \mathbb{R}^d$ can be derived from the robot's state $\mathbf{x}$ through a potentially non-linear projection function $\mathbf{c} \in \mathcal{X} \to \mathbb{R}^d$;

$$\mathbf{p} = \mathbf{c}(\mathbf{x}), \tag{4}$$

and that the geometry $\mathcal{R}$ of the robot is the robot's smallest enclosing sphere with radius r centered at its reference

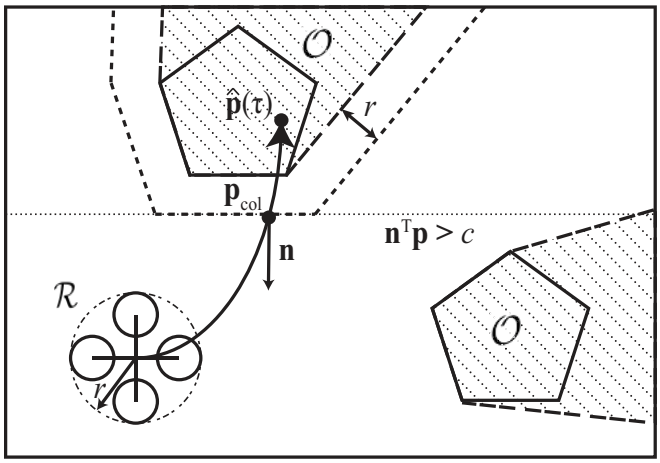

Fig. 2. A schematic picture illustrating our approach. The future trajectory of the robot is extrapolated given the current state of the robot and current operator input, and is checked for collisions against the obstacles $\mathcal{O}$, which include the "shadows" of the obstacles that cannot be seen from the vantage point of the robot. If a collision is imminent within the time horizon τ, a halfspace is defined tangent to the point of collision that induces a constraint for the change in control input $\Delta\mathbf{u}$.

point (such that its geometry is invariant to orientation). Let $\mathcal{R}(\mathbf{p}) \subset \mathbb{R}^d$ denote the subset of the workspace (a sphere) occupied by the robot when it is at position $\mathbf{p}$.

Secondly, given the current state $\mathbf{x}$ of the robot, and a change $\Delta\mathbf{u}$ to the operator's control input $\mathbf{u}$, let the position of the robot at time t be approximated by the first-order Taylor expansion:

$$\mathbf{p}(t, \Delta\mathbf{u}) \approx \hat{\mathbf{p}}(t) + G(t)\Delta\mathbf{u} \tag{5}$$

where

$$\hat{\mathbf{p}}(t) = \mathbf{c}(\mathbf{g}(\mathbf{x}, \mathbf{u}, t)), \quad G(t) = \frac{\partial(\mathbf{c} \circ \mathbf{g})}{\partial \mathbf{u}}(\mathbf{x}, \mathbf{u}, t). \tag{6}$$

In addition, we make an implicit assumption about the local linearity of the future trajectory of the robot, such that if the robot is collision-free at the time horizon τ with respect to an appropriately chosen convex subset of the free workspace ($\mathbb{R}^d \setminus \mathcal{O}$), then it is also collision-free for all times $t \in [0, \tau]$. This is reasonable as long as τ is not too large, and the convex subset of the free workspace includes the current position of the robot.

B. Basic Approach

Our approach is as follows: given the current state $\mathbf{x}$ and the current operator's control input $\mathbf{u}$, the (exact) future positions of the robot are given by the function $\hat{\mathbf{p}}$ of Eq. (5). If for all $t \in [0, \tau]$ the robot does not collide with any obstacle, i.e. $\forall t \in [0, \tau] :: \mathcal{R}(\hat{\mathbf{p}}(t)) \cap \mathcal{O} = \emptyset$, then the operator's control input need not be changed: $\Delta\mathbf{u} = \mathbf{0}$.

If a collision is encountered, let $\mathbf{p}_{\text{col}}$ be the first position of the robot at which it is in collision with the obstacles:

$$\mathbf{p}_{\text{col}} = \hat{\mathbf{p}}(\min\{t \in [0, \tau] \mid \mathcal{R}(\hat{\mathbf{p}}(t)) \cap \mathcal{O} \neq \emptyset\}), \tag{7}$$

and let $\mathbf{n}$ be the unit normal vector to $\mathcal{O}$ (pointing into the free workspace) at the point of collision (see Fig. 2). We now define the following linear constraint on the position $\mathbf{p}(\tau, \Delta\mathbf{u})$ of the robot at the time-horizon τ:

$$\mathbf{n}^T\mathbf{p}(\tau, \Delta\mathbf{u}) > c, \tag{8}$$

where $c = \mathbf{n}^T\mathbf{p}_{\text{col}}$. Eq. (8) defines a halfspace that gives a (rough) convex approximation of the local free space. Substituting Eq. (5) transforms it into a constraint on $\Delta\mathbf{u}$:

$$\mathbf{a}^T\Delta\mathbf{u} > b, \tag{9}$$

where $\mathbf{a}^T = \mathbf{n}^T G(\tau)$ and $b = c - \mathbf{n}^T\hat{\mathbf{p}}(\tau)$. Replacing the complex constraint of Eq. (3) by the simple constraint of Eq. (9) then gives an closed-form solution for $\Delta\mathbf{u}$:

$$\Delta\mathbf{u} = bR^{-1}\mathbf{a}/(\mathbf{a}^T R^{-1}\mathbf{a}). \tag{10}$$

C. Iteration

The new control input $\mathbf{u} + \Delta\mathbf{u}$ does not guarantee that the future trajectory of the robot is collision-free with respect to all obstacles for all $t \in [0, \tau]$, in particular near convex corners and edges of the free-space (see Fig. 3). Therefore, we repeat the above approach in an iterative fashion.

Let the change in control input computed in Eq. (10) be denoted $\Delta\mathbf{u}_1$, and let this have been iteration 1 of the algorithm. Then, in iteration i, we extrapolate the trajectory for control input $\mathbf{u} + \Delta\mathbf{u}_{i-1}$ and search for the first collision. This defines a linear constraint $\mathbf{a}_i^T\Delta\mathbf{u} > b_i$ on the change in control input similar as in Eq. (9). We now solve an optimization problem with respect to i constraints:

$$\begin{aligned} &\text{minimize:} && \Delta\mathbf{u}^T R\Delta\mathbf{u} \\ &\text{subject to:} && \textstyle\bigcap_{j=1}^{i}\{\mathbf{a}_j^T\Delta\mathbf{u} > b_j\} \end{aligned} \tag{11}$$

In each iteration a constraint is added to the convex optimization problem. In our implementation we let the number of iterations, and hence the number of constraints, be maximized at d (the dimension of the workspace) to account for corners of the free space in d dimensions (see Fig. 3 for an illustration where $d = 2$). The control input $\mathbf{u} + \Delta\mathbf{u}$, where $\Delta\mathbf{u}$ is the change in control input computed in the last iteration, is then applied to the robot. Note that the above approach is repeated in each control cycle of the robot.

The iterative inclusion of constraints in this optimization problem aligns well with the LP-type algorithm of [31] that can solve such low-dimensional convex optimization problems in $O(i)$ expected time by considering the constraints in an iterative fashion (i being the number of constraints). The dimension of our optimization problem equals the dimension m of the control input $\Delta\mathbf{u}$, which is typically equal to the dimension d of the workspace. The fact that we maximize the number of iterations at d then ensures that the convex optimization problem is always feasible.

V. Implementation and Experiments

In this section we describe the implementation of our approach on a physical quadrotor helicopter in a laboratory environment, and qualitatively discuss experimental results. We refer the reader to the video accompanying this paper and to **http://arl.cs.utah.edu/research/aca** for videos of our experiments.

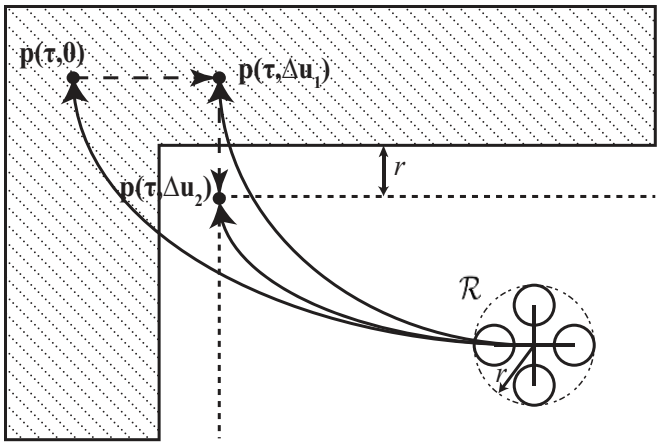

Fig. 3. A schematic illustration of our iterative optimization approach near convex corners of the free workspace. The future trajectory given the operator's control input collides with the left wall, so a change $\Delta\mathbf{u}_1$ is computed that avoids collision with the left wall. However, this change in control input will let the robot collide with the top wall, so a change in control input $\Delta\mathbf{u}_2$ is computed in an optimization problem with two constraints. The number of iterations, and hence the number of constraints, is maximized at d, the dimension of the workspace, which in this example is $d = 2$. The change in control input $\Delta\mathbf{u}_2$ that is computed in the last iteration is then applied to the robot.

A. Quadrotor Dynamics

We used a Parrot AR.Drone 2.0 quadrotor in our experiments. Its state $\mathbf{x} = [\mathbf{p}^T, \mathbf{v}^T, \mathbf{r}^T, \mathbf{w}^T]^T \in \mathcal{X}$ is 12-dimensional, and consists of its position $\mathbf{p} \in \mathbb{R}^3$ (m), its velocity $\mathbf{v} \in \mathbb{R}^3$ (m/s), its orientation $\mathbf{r} \in \mathbb{R}^3$ (rad) (rotation about axis $\mathbf{r}/\|\mathbf{r}\|$ by angle $\|\mathbf{r}\|$), and angular velocity $\mathbf{w} \in \mathbb{R}^3$ (rad/s). Its control input $\mathbf{u} = [u_1, u_2, u_3]^T \in \mathcal{U}$ is 3-dimensional and consists of desired vertical velocity u_1 (m/s), desired roll u_2 (rad), and desired pitch u_3 (rad). The AR.Drone also allows input for its yaw, but since this is a redundant degree-of-freedom, we implicitly hold this input constant at zero. The actual internal dynamics of the Parrot AR Drone 2.0 are not completely known to us, but we hypothesize that its dynamics (i.e. the function $\mathbf{f}$ of Eq. (1)) are approximated well by:

$$\dot{\mathbf{p}} = \mathbf{v} \tag{12}$$

$$\dot{\mathbf{v}} = -k_{\text{drag}}\mathbf{v} + \exp([\mathbf{r}])[0, 0, k_{p1}(u_1 - v_z)]^T \tag{13}$$

$$\dot{\mathbf{r}} = \mathbf{w} \tag{14}$$

$$\dot{\mathbf{w}} = \begin{bmatrix} k_{p2}(u_2 - r_x) - k_d w_x \\ k_{p2}(u_3 - r_y) - k_d w_y \\ -k_{p3} w_z \end{bmatrix}, \tag{15}$$

where $[\mathbf{r}]$ represents the skew-symmetric cross-product matrix of $\mathbf{r}$, and k_{drag}, k_{p1}, k_{p2}, k_{p3}, and k_d are coefficients and gains whose values were determined using standard parameter estimation techniques. Note that we use a simple yaw controller on the quadrotor to ensure that its yaw remains zero. We set the AR.Drone to an aggressive flight mode that allows for roll and pitch angles of up to 0.4rad.

B. Experiment Setup and Implementation Details

We performed our physical experiments within a laboratory environment equipped with an OptiTrack V100:R2 motion capture system. All computations were performed on an off-board computer with an Intel Core i7 3.4GHz processor with 8GB RAM. We implemented our approach within the Robot Operating System (ROS) framework, and ran it with a control cycle of 50 Hz. Controls were sent to the vehicle over a wireless WiFi connection, and operator controls were received via USB Xbox controller. The current state of the vehicle was continually estimated using a Kalman filter that obtains position information from the motion capture system through the Motive software interface, and obtains information regarding velocity, orientation, and angular velocity from sensors on-board the AR.Drone.

The obstacle environment was predefined for each experiment and represented in the form of a set of oriented triangular facets. These triangles model the true obstacles offset along their normals by the radius r of the bounding sphere of the robot, such that the robot itself can be treated as a point. Given the current state $\mathbf{x}$ of the quadrotor from the Kalman filter, and an operator control input $\mathbf{u}$ from the Xbox controller, the future trajectory of the robot was estimated by integrating the function $\mathbf{f}$ of Eq. (1) (as specified in Section V-A) forward in time using 4th-order Runge-Kutta integration with small time increments. Between each increment, the trajectory of the robot was approximated by a straight-line segment, which was checked for intersection with each of the triangular obstacle facets [32]. Halfspace constraints induced by intersections, as in Eq. (8), were slightly offset by a safety margin (i.e. c was increased) to prevent the robot from coming too close to obstacles. The matrix $G(\tau)$ was approximated using numerical differentiation. We used the iterative method outlined in Section IV-C along with Eq.(10) to compute $\Delta\mathbf{u}$ in each control cycle given the halfspace constraints.

C. Setting the Time Horizon τ

The time horizon τ, i.e. the amount of time the algorithm looks ahead, is an important parameter of our approach. Setting it too low may cause collisions, since the robot is given too little time to avoid them once they are predicted. Setting it too high may cause too conservative behavior, in which the algorithm constantly overrides the operator, even if there is enough time to avoid a potential future collision.

To quantitatively assess the effect of the value of τ on our method, we performed an experiment in which the robot was repeatedly flown along a path perpendicular to a wall for varying values of τ. The virtual wall was defined by a single constant halfspace constraint of the form of Eq. (8), with respect to which a change in control input $\Delta\mathbf{u}$ was computed in each control cycle. Fig. 4 shows the position of the robot along the perpendicular axis of the virtual wall as a function of time for multiple values of τ. It can be seen that setting τ too small ($\tau = 0.75$s) results in an under-damped response, which causes the vehicle to oscillate about the position of the virtual wall. Resulting in unsafe flight with undesirable oscillatory behavior. Setting τ too large ($\tau = 1.75$s) results in an over-damped response, which corresponds to safe but overly conservative behavior. The value of τ that results in a

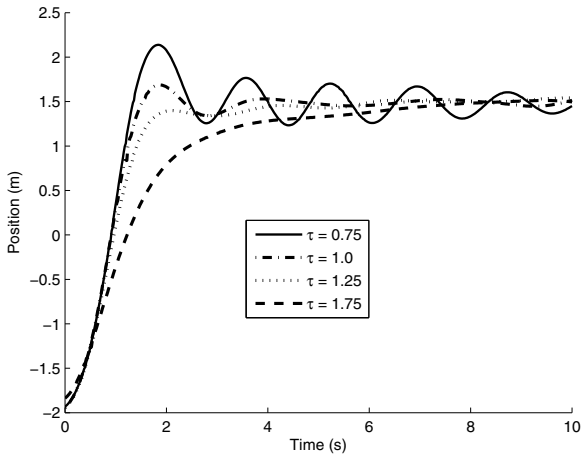

Fig. 4. A diagram showing the experimental results from varying the time horizon τ of our collision avoidance approach for the Parrot AR.Drone 2.0 quadrotor. The vehicle was flown straight towards a virtual wall for various values of τ. The graphs shows the position of the robot along the axis perpendicular to the virtual wall as a function of time during each of the flights for multiple values of the time horizon τ. As the time horizon increases the system's response changes from underdamped ($\tau = 0.75$s), to critically damped ($\tau = 1.25$s), to overdamped ($\tau = 1.75$s).

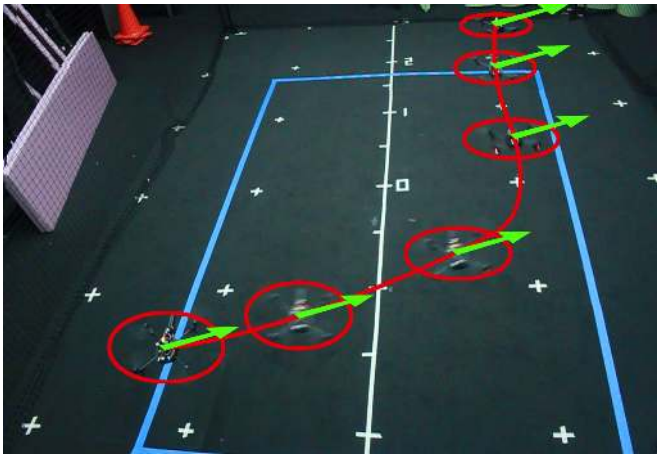

Fig. 5. A time-lapsed picture showing the behavior of our approach when the operator attempts to fly the quadrotor into a wall (blue lines) at an oblique angle. Our approach only uses the component of the operator's control input (green arrows) that is parallel to the obstacle, and ignores the component of the control input that is perpendicular and directed into the obstacle, in order to prevent collisions from happening while at the same time maintaining the operator's intent as closely as possible. We refer the reader to the video accompanying this paper and to **http://arl.cs.utah.edu/research/aca** for videos of our experiments.

critically damped response is most desirable. Based on this experiment we estimate this value to be about $\tau = 1.25$s, and we used this value in all subsequent experiments.

It is important to note that the optimal time horizon τ will be different for each robotic system, as it depends on the responsiveness of the dynamics of the vehicle. Hence, the above experiment must be repeated if our approach were implemented on a different vehicle.

D. Experimental Results

We experimented in the environment of Figs. 1, 5, and 6, where in all cases the obstacles included four (virtual) walls (as indicated by the blue lines in the figure), a ceiling, and a floor. In the first experiment (see the video accompanying this paper or videos on **http://arl.cs.utah.edu/research/aca**), we let the quadrotor be flown along the boundaries of the domain. This experiment highlights our approach's ability to deal with convex edges and corners of the free space, even when the operator attempts to crash the vehicle into each of the four corners. As can be seen, our approach prevented this from happening, and slows the quadrotor down just before it would hit a wall. In the second experiment (see Fig. 5) the operator controls the quadrotor to fly towards a wall at an oblique angle. Before the quadrotor hits the wall, our approach deflects the movement of the vehicle such that it flies parallel to the wall rather than into it. This experiment highlights the capability of our approach to let the operator use the obstacles for compliant flight; our approach only uses the component of the operator's control input that is parallel to the obstacle, and ignores the component of the control input that is perpendicular and directed into the obstacle. In the third experiment (see Fig. 6) we added an obstacle to the environment that protrudes from one of the walls. When the operator flies the quadrotor along the wall, our approach makes sure that the obstacle is avoided by letting making an evasive maneuver before returning to a flight trajectory parallel to the wall. The fourth experiment is similar (see Fig. 1), but here the obstacle protrudes from the floor. In this case the operator attempted to fly the quadrotor into the floor and into the obstacle, but our approach prevented collisions by increasing the vehicle's altitude so that it safely flew over the obstacle. These experiments show our approach's ability to handle concave edges and corners of the free space. The video accompanying this paper also shows an experiment where the vehicle is flown head-on to one of the obstacle edges. In all experiments, the operator was unable to crash the vehicle, even when deliberately attempting to do so.

VI. Conclusions and Future Work

In this paper, we presented an approach for automatic collision avoidance for manually tele-operated unmanned aerial robots. Our experiments on a physical quadrotor show that our approach is capable of avoiding collisions with obstacles while at the same time following the operator's control input as closely as possible. This leads to an intuitive control interface that allows the operator to focus his attention on global decision-making and rely on the system to avoid collisions with obstacles and walls. While we implemented our approach for quadrotor helicopters, we conceptually developed our approach for general systems with non-linear dynamics. In future work we will explore the applicability of our approach on other types of tele-operated (mobile) robots.

Our approach has conceptually been designed such that all computations can be performed on-board the vehicle. However, our experiments were performed in a laboratory environment whose obstacle geometry was given. To make our approach applicable in unknown, indoor environments, our approach would need to be augmented with some form

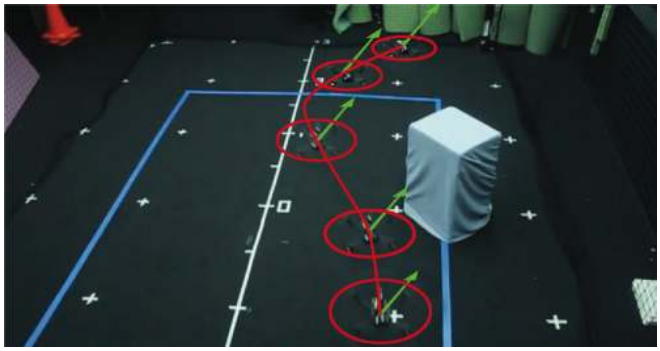

Fig. 6. A time-lapsed picture showing the behavior of our approach when the operator attempts (green arrows) to fly the quadrotor along a wall (blue lines) out of which an obstacle protrudes. Our approach makes sure that the obstacle is avoided by letting the quadrotor make an evasive maneuver before returning to a trajectory parallel to the wall. We refer the reader to the video accompanying this paper and to **http://arl.cs.utah.edu/research/aca** for videos of our experiments.

of on-board SLAM [15], [16], [33]. We plan to do this in future work. Other possible future extensions include combining our approach with a haptic feedback mechanism, and enabling our approach to avoid collisions with moving obstacles and other robots.

REFERENCES

[1] J. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, "Quadrotor using minimal sensing for autonomous indoor flight," in *Proc. of the European Micro Air Vehicle Conference and Flight Competition*, 2007.

[2] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a gps-denied environment," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 1814–1820.

[3] J. Mendes and R. Ventura, "Assisted teleoperation of quadcopters using obstacle avoidance," *Journal of Automation, Mobile Robotics & Intelligent Systems*, vol. 7, no. 1, 2013.

[4] L. Hull, "Drone makes first uk arrest as police catch car thief hiding under bushes," *Daily Mail*, vol. 12, 2010.

[5] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 4, no. 3, pp. 143–153, 2003.

[6] O. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen, "Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations," *Vehicle System Dynamics*, vol. 44, no. 7, pp. 569–590, 2006.

[7] J. C. McCall and M. M. Trivedi, "Driver behavior and situation aware brake assistance for intelligent vehicles," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 374–387, 2007.

[8] J. Cao, H. Liu, P. Li, and D. J. Brown, "State of the art in vehicle active suspension adaptive control systems based on intelligent methodologies," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, no. 3, pp. 392–405, 2008.

[9] L. B. Rosenberg, "Virtual fixtures: Perceptual tools for telerobotic manipulation," in *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*, 1993, pp. 76–82.

[10] P. Marayong, M. Li, A. M. Okamura, and G. D. Hager, "Spatial motion constraints: Theory and demonstrations for robot guidance using virtual fixtures," in *Robotics and Automation (ICRA), 2003 IEEE International Conference on*, vol. 2, 2003, pp. 1954–1959.

[11] A. Bettini, P. Marayong, S. Lang, A. M. Okamura, and G. D. Hager, "Vision-assisted control for manipulation using virtual fixtures," *Robotics, IEEE Transactions on*, vol. 20, no. 6, pp. 953–966, 2004.

[12] M. Dewan, P. Marayong, A. M. Okamura, and G. D. Hager, "Vision-based assistance for ophthalmic micro-surgery," in *Proc. Medical Image Computing and Computer-Assisted Intervention*, 2004, pp. 49–57.

[13] J. J. Abbott, P. Marayong, and A. M. Okamura, "Haptic virtual fixtures for robot-assisted manipulation," in *Proc. Int. Symp. on Robotics Research*, 2007, pp. 49–64.

[14] T. L. Gibo, L. N. Verner, D. D. Yuh, and A. M. Okamura, "Design considerations and human-machine performance of moving virtual fixtures," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 2009, pp. 671–676.

[15] S. Thrun, M. Diel, and D. Hähnel, "Scan alignment and 3-d surface modeling with a helicopter platform," in *Field and Service Robotics*, 2006, pp. 287–297.

[16] A. Bry, A. Bachrach, and N. Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2012)*, St Paul, MN, 2012.

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

[18] 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.

[19] M. Nieuwenhuisen, D. Droeschel, J. Schneider, D. Holz, T. Labe, and S. Behnke, "Multimodal obstacle detection and collision avoidance for micro aerial vehicles," in *Mobile Robots (ECMR), 2013 European Conference on*. IEEE, 2013, pp. 7–12.

[20] 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.

[21] P. Saranrittichai, N. Niparnan, and A. Sudsang, "Robust local obstacle avoidance for mobile robot based on dynamic window approach," in *Int. Conf. on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, 2013.

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

[23] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen, "Collision avoidance under bounded localization uncertainty," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2012.

[24] J. van den Berg, D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha, "Lqg-obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 346–353.

[25] T. Fraichard and H. Asama, "Inevitable collision states. a step towards safer robots?" in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, 2003, pp. 388–393.

[26] G. Ducard and R. D'Andrea, "Autonomous quadrotor flight using a vision system and accommodating frames misalignment," in *Industrial embedded systems, 2009. SIES'09. IEEE international symposium on*, 2009, pp. 261–264.

[27] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 2009, pp. 2878–2883.

[28] D. Bareiss and J. van den Berg, "Reciprocal collision avoidance for robots with linear dynamics using lqr-obstacles," in *IEEE Int. Conf. Robotics and Automation*, 2013.

[29] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 477–483.

[30] A. M. Brandt and M. B. Colton, "Haptic collision avoidance for a remotely operated quadrotor uav in indoor environments," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, 2010, pp. 2724–2731.

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

[32] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," *Journal of graphics tools*, vol. 2, no. 1, pp. 21–28, 1997.

[33] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 3, pp. 229–241, 2001.