MODEL-BASED COLLISION AVOIDANCE FOR DYNAMIC SINGLE- AND MULTI-ROBOT SYSTEMS: THEORY AND APPLICATION IN GROUND AND AERIAL ROBOTS

by

Daman F. Bareiss

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Mechanical Engineering

The University of Utah

August 2016

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Daman F. Bareiss**

has been approved by the following supervisory committee members:

Kam K. Leang	, Chair	**05/13/2016** Date Approved
Jur Van Den Berg	, Member	**05/17/2016** Date Approved
Jake J. Abbott	, Member	**05/13/2016** Date Approved
Katia N. Estabridis	, Member	**05/13/2016** Date Approved
David E. Johnson	, Member	**05/13/2016** Date Approved

and by **Timothy A. Ameel**, Chair/Dean of

the Department/College/School of **Mechanical Engineering**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

This dissertation solves the collision avoidance problem for single- and multi-robot systems where dynamic effects are significant. In many robotic systems (e.g., highly maneuverable and agile unmanned aerial vehicles) the dynamics cannot be ignored and collision avoidance schemes based on kinematic models can result in collisions or provide limited performance, especially at high operating speeds. Herein, real-time, model-based collision avoidance algorithms that explicitly consider the robots' dynamics and perform real-time input changes to alter the trajectory and steer the robot away from potential collisions are developed, implemented, and verified in simulations and physical experiments. Such algorithms are critical in applications where a high degree of autonomy and performance are needed, for example in robot-assisted first response where aerial and/or mobile ground robots are required to maneuver quickly through cluttered and dangerous environments in search of survivors. Firstly, the research extends reciprocal collision avoidance to robots with dynamics by unifying previous approaches to reciprocal collision avoidance under a single, generalized representation using control obstacles. In fact, it is shown how velocity obstacles, acceleration velocity obstacles, continuous control obstacles, and linear quadratic regulator (LQR)-obstacles are special instances of the generalized framework. Furthermore, an extension of control obstacles to general reciprocal collision avoidance for nonlinear, nonhomogeneous systems where the robots may have different state spaces and different nonlinear equations of motion from one another is described. Both simulations and physical experiments are provided for a combination of differential-drive, differential-drive with a trailer, and car-like robots to demonstrate that the approach is capable of letting a nonhomogeneous group of robots with nonlinear equations of motion safely avoid collisions at real-time computation rates. Secondly, the research develops a stochastic collision avoidance algorithm for a tele-operated unmanned aerial vehicle (UAV) that considers uncertainty in the robot's dynamics model and the obstacles' position as measured from sensors. The model-based automatic collision avoidance algorithm is implemented on a custom-designed quadcopter UAV system with on-board computation and the sensor data are processed using a split-and-merge segmentation algorithm and an approximate

Minkowski difference. Flight tests are conducted to validate the algorithm's capabilities for providing tele-operated collision-free operation. Finally, a set of human subject studies are performed to quantitatively compare the performance between the model-based algorithm, the basic risk field algorithm (a variant on potential field), and full manual control. The results show that the model-based algorithm performs significantly better than manual control in both the number of collisions and the UAVs average speed, both of which are extremely vital, for example, for UAV-assisted search and rescue applications. Compared to the potential-field-based algorithm, the model-based algorithm allowed the pilot to operate the UAV with higher average speeds.

For my wife Crystal and my son Devon.

Thank you for putting up with the life of a graduate student.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This dissertation is the result of the support, encouragement, and mentorship I've received during my time as a graduate student at the University of Utah. First, I want to thank my advisors, Dr. Jur van den Berg for helping me begin the process of becoming a successful researcher and exposing me to the world of algorithms and motion planning as a mechanical engineering student, and Dr. Kam K. Leang for helping me continue to make accomplishments and complete my studies. Second, I want to thank the rest of my committee members, Drs. Abbott, Estabridis, and Johnson, for their valuable technical insights. Third, I am grateful to the members of the DARC Lab and the ARL for their conversations and insights. Finally, I want to thank my wife Crystal and my family for their patience during this endeavor.

This work was funded by several sources: the National Science Foundation through an Integrative Graduate Education and Research Traineeship (IGERT Grant No. 0654414), the National Science Foundation through the Partnership for Innovation Program (Grant No. 1430328), the Achievement Rewards for College Scientists (ARCS) Foundation, and the University of Utah Graduate Research Fellowship.

CHAPTER 1

INTRODUCTION

1.1 Motivation and Research Focus

The number of civil and commercial applications for ground, water, and aerial robots has grown significantly over the past few decades due to advances in their hardware, as well as their design, manufacturing, and the development of innovative algorithms for artificial intelligence and autonomy. Applications include mapping [1], search and rescue [2], [3], precision farming [4], space exploration [5], traffic management [6], environmental monitoring [7], and even entertainment [8]. Despite recent advancements, one of the major challenges in mobile ground, water, and aerial robotic systems is the task of avoiding collisions with surrounding obstacles as robots travel and navigate through complex and unstructured environments. Therefore, reliable, robust, and effective automatic collision avoidance algorithms are critical in applications where a high degree of autonomy and performance are needed, for example in robot-assisted first response where aerial and/or mobile ground robots are required to maneuver quickly through cluttered and dangerous environments in search of survivors following a natural disaster. Motivated by this need, the main focus of this dissertation is solving the collision avoidance problem for single- and multi-robot systems where dynamic effects are significant. In fact, the dynamic effects inherent in highly maneuverable and agile systems such as unmanned aerial vehicles cannot be ignored when developing collision avoidance schemes. For instance, collision avoidance schemes based on kinematic models can result in collisions or provide limited performance, and thus such systems tend to operate more conservatively compared to schemes that consider the robot's dynamics. Herein, real-time, model-based collision avoidance algorithms that explicitly consider the robots' dynamics and perform real-time input changes to alter the trajectory and steer the robot away from potential collisions are developed, implemented, and verified in simulations and physical experiments. Developing innovative collision avoidance algorithms that enable robots (including teams of robots) to better maneuver through challenging terrain with greater efficiency, speed, and safety helps to broaden their application space.

Collision avoidance algorithms can typically be classified as either a global or a local method. Global methods work on a knowledge of all the obstacles in the workspace and typically focus on optimizing some criterion and performing path refinement [9]–[13]. Other methods perform sampling provided knowledge of the environment and can also perform rapid replanning in the presence of moving obstacles [14]–[17]. Local methods are preferable in situations where the environment is not well characterized or known *a priori*, and these algorithms typically run in real-time and avoid collisions with respect to a limited, local obstacle definition, eliminating the need for a global obstacle map to be provided or estimated. However, many early collision avoidance methods are developed under the assumption that the robots' dynamics can be ignored [18]–[29]. This is not a valid assumption for highly dynamic robots, such as quadcopters or fixed-wing aircraft. When these robots are operated at high speeds where their dynamic effects are not negligible, the collision avoidance methods can still have collisions occur from approximating these robots as kinematic agents. To fully utilize these robots' capabilities, their dynamics should be considered in avoiding collisions.

1.2 Research Objectives

The outcomes of this dissertation will advance the state-of-the-art for local collision avoidance through the development of local collision avoidance algorithms that incorporate the robots' potentially nonlinear dynamics and perform real-time input changes to alter the robots' trajectories to avoid collisions. Specifically, algorithms for both multi- and single-robot systems are developed through the following objectives:

1.2.1 Objective 1: Extend Reciprocal Collision Avoidance to Robots with Dynamics

This objective focuses on creating a reciprocal collision avoidance algorithm for nonhomogeneous systems of robots with potentially nonlinear dynamics [30]. This objective was completed by performing three tasks that are described as follows:

1.2.1.1 Develop control obstacles for linear, homogeneous systems

First, Control Obstacles for linear, homogeneous systems are developed. These systems have been used in previous reciprocal collision avoidance algorithms. However, the previous methods avoid collisions by selecting a new *total* relative input. Control Obstacles are developed in terms of a *change* in relative input. This difference in the input representation is

significant for the generalization of previous methods and the extension to nonhomogeneous systems with potentially nonlinear dynamics.

1.2.1.2 Generalize previous reciprocal collision avoidance algorithms

This task generalized previous reciprocal collision avoidance algorithms by showing how their robot models are represented by Control Obstacles. It is shown how velocity obstacles [20], acceleration velocity obstacles [31], continuous control obstacles [32], and linear quadratic regulator (LQR)-obstacles [33] are special instances of the generalized framework. These systems each use different models for the robots with varying complexities, ranging from purely kinematic to general, linear dynamics. All of them can be represented as Control Obstacles with the proper selection of terms as shown.

1.2.1.3 Extend control obstacles to nonlinear systems

This task extended Control Obstacles for use with nonhomogeneous systems of robots with possibly nonlinear dynamics. By an approximation of the robots' trajectories through a first-order Taylor expansion about their current inputs, multiple robots with different, nonlinear dynamics can avoid collisions.

1.2.2 Objective 2: Develop a Feedforward Collision Avoidance Algorithm for a Tele-Operated Robot

A feedforward (model-based) collision avoidance algorithm that performs well in real-time operation for tele-operation by a human operator is developed [34]–[37]. This objective is completed by performing three tasks that are described as follows:

1.2.2.1 Develop stochastic collision avoidance algorithm

This task completed the theoretical development of the algorithm for performing automatic collision avoidance for tele-operated unmanned aerial vehicles [34], [35]. This algorithm explicitly considers uncertainty in the motion model of the robot for the feedforward prediction of the trajectory as well as the uncertainty in the location of the obstacles.

1.2.2.2 Implement in real time with onboard sensing

The theoretical developments from [34], [35] are implemented on a custom-designed quadcopter UAV system with on-board computation and sensing capabilities [36]. Sensor data are processed using a split-and-merge segmentation algorithm with an approximate Minkowski difference. Flight tests are conducted to validate the algorithm's capabilities.

1.2.2.3 Study impact of collision avoidance algorithms on pilot performance

The final task presents a set of human-subject studies that are performed to quantitatively compare the performance of pilots between the model-based algorithm of Obj. 1.2.2 and the basic risk field (a variation of the potential field [24]) and full manual control [37]. The human-subject study has been reviewed and approved by the University of Utah Institutional Review Board.

Additional details related to the two objectives are presented next.

1.3 Multirobot Systems: Reciprocal Collision Avoidance

When multiple robots share a common workspace, collisions must be avoided between the robots as well as between the obstacles in the environment. Early approaches involved using algorithms developed for single-robot collision avoidance where each robot assumes every other robot is a passive obstacle in the environment. However, such approaches can be insufficient when the robot encounters other robots that also actively make decisions based on their surroundings: considering them as moving obstacles overlooks the fact that they react to the robot in the same way as the robot reacts to them, and inherently causes suboptimal and oscillatory motion [38], [39].

This has lead to the development of reciprocal collision avoidance techniques, which specifically account for the reactive nature of the other robots without relying on coordination or communication among robots. The earliest approaches were direct extensions of Velocity Obstacles [20], in which each robot is given half the responsibility of avoiding pairwise collisions [39], [40]. Since this approach only applies to robots with very simple equations of motion that allow the robots to change their velocity instantaneously, most subsequent research on the topic has focused on extending the approach to robots with more complex dynamics constraints, such as differential-drive [41], [42], car-like [43], double-integrator [31], [44], arbitrary-degree integrator [32], and linear quadratic regulator (LQR)-controlled [33] robots. A major limitation of these approaches, though, is that all robots are assumed to have exactly the same equations of motion, i.e., they apply to *homogeneous* systems only. Moreover, in all of these approaches the assumed equations of motion are *linear*, as these approaches rely on the ability to express the *relative* motion of pairs of robots in terms of the *relative* control input (i.e., the difference between the control inputs) of the robots. Hence, they do not apply to nonhomogeneous or nonlinear systems, where robots have different and/or nonlinear equations of motion, which limits their applicability on real-world robots. The reciprocal collision avoidance algorithm in this dissertation advances the previous work

by developing Control Obstacles for systems of robots that can have different, potentially nonlinear dynamics [30]. Control Obstacles work in the space of *changes* in input while Velocity Obstacles work in the space of absolute velocities. These Control Obstacles are shown to be a generalization of the previous reciprocal collision avoidance methods and their extension to nonlinear systems is shown in both simulation and real-world experiments.

1.4 Single-Robot Systems: Automatic Collision Avoidance for Teleoperation

In the aforementioned applications such as search and rescue or inspection of hazardous locations, a pilot or operator of a robot such as an unmanned aerial vehicle (UAV) 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 limited information, such as a live camera-feed from the vehicle.

To aid the human operator in such tasks, it would be beneficial to incorporate an algorithm that lets the vehicle automatically perform collision avoidance, such that the operator can focus their attention on 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 [45]–[48], this dissertation focuses on approaches designed specifically so that the operator can *rely* on the collision avoidance system. Our system ensures that collisions are avoided while maintaining 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 [49] that are commonly used in surgical robotics [50]–[54]. This dissertation develops a feedforward collision avoidance algorithm for a tele-operated robot [35]–[37]. Given some desired input, the potentially nonlinear dynamics of the robot can be propagated forward in time to estimate the trajectory. If a collision is predicted to occur given the estimated trajectory, the user's input is updated in real time to avoid collision. The algorithm is developed to consider both the uncertainty in the motion model of the robot and the uncertainty in the sensing of the obstacles' positions through a simplified Gaussian representation [35]. The approach is validated with real-world implementation using on-board sensing and computation [36]. Studies are also performed in simulation to quantify how a pilot's performance is improved when using the algorithm [37].

1.5 Organization

This dissertation is organized as follows. Chapter 2 describes the work that extends reciprocal collision avoidance to robots with dynamics. Chapter 3 presents the theoretical work for the stochastic automatic collision avoidance algorithm for tele-operated UAVs and shows simulation results. Chapter 4 describes the real-time, on-board implementation and experimental verification of the stochastic automatic collision avoidance algorithm for a custom-designed tele-operated UAV. The experimental system consists of on-board computation and sensing, and flight experiments are presented to demonstrate the performance of the algorithm. Chapter 5 deals with human subject studies to determine quantitatively the impact on UAV-pilot performance when using the automatic collision algorithm compared to full manual control and a potential-field-based method. Discussions and future considerations are presented in Chapter 6, and finally conclusions are presented in Chapter 7.

1.6 References

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

[2] S. Waharte and N. Trigoni, "Supporting search and rescue operations with UAVs," in *IEEE Int. Conf. on Emerging Security Technologies*, 2010, pp. 142–147.

[3] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, "Supporting wilderness search and rescue using camera equipped mini uav," *J. of Field Robotics*, vol. 25, no. 1-2, pp. 89–110, 2008.

[4] C. Zhang and J. M. Kovas, "The application of small unmanned aerial systems for precision agriculture: a review," *13*, pp. 693–712, 2012.

[5] G. A. Landis, "Robots and humans: synergy in planetary exploration," *Acta astronautica*, vol. 55, no. 12, pp. 985–990, 2004.

[6] P. S. Lin, L. Hagen, K. Valavanis, and H. Zhou, "Vision of unmanned aerial vehicle (uav) based traffic management for incidents and emergencies," in *12th World Congress on Intelligent Transport Systems*, 2005, pp. 1–12.

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

[8] H. Yoshimoto, K. Jo, and K. Hori, "Toward entertainment blimps for everyone by everyone," in *Proceedings of the seventh ACM conference on creativity and cognition*, 2009, pp. 445 –446.

[9] A. Bry and N. Roy, "Rapidly exploring random belief trees for motion planning under uncertainty," in *IEEE Int. Conf. on Robotics and Automation*, 2011.

[10] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: stochastic trajectory optimization for motion planning," in *IEEE Int. Conf. on Robotics and Automation*, 2011.

[11] J. Muller and G. S. Sukhatme, "Risk-aware trajectory generation with application to safe quadrotor landing," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 3642–3648.

[12] S. Prentice and N. Roy, "The belief roadmap: efficient planning in belief space by factoring the covariance," *Int. J. of Robotics Research*, vol. 28, no. 11-12, pp. 1448–1465, 2009.

[13] J. van den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: optimized path planning for robots with motion uncertainty and imperfect state information," *Int. J. of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[14] D. Ferguson, A. Kalra, and A. Stentz, "Replanning with rrts," in *IEEE Int. Conf. on Robotics and Automation*, 2006.

[15] R. Gayle, A. Sud, M. C. Lin, and D. Manocha, "Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007.

[16] I. A. Sucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Trans. on Robotics*, vol. 28, no. 1, pp. 116–131, 2011.

[17] K. E. Bekris and L. E. Kavraki, "Greedy but safe replanning under kinodynamic constraints," in *IEEE Int. Conf. on Robotics and Automation*, 2007.

[18] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[19] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: a collision cone approach," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 28, no. 5, pp. 562–574, 1998.

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

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

[22] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid," in *IEEE Int. Conf. on Robotics and Automation*, 2007.

[23] L. He and J. van den Berg, "Meso-scale planning for multi-agent navigation," in *IEEE Int. Conf. on Robotics and Automation*, 2013.

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

[25] B. Kluge and E. Prassler, "Recursive probabilistic velocity obstacles for reflective navigation," in *Field and Service Robotics*. Springer, 2006, vol. 24, pp. 71–79, NEED ILL.

[26] J. Minguez and L. Montano, "Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios," in *IEEE Int. Conf. on Robotics and Automation*, 2004.

[27] M. Mujahed, H. Jaddu, D. Fischer, and B. Mertsching, "Tangential closest gap based (tcg) reactive obstacle avoidance navigation for cluttered environments," in *IEEE Int. Symp. on Safety, Security, and Rescue Robots*, 2013.

[28] A. K. Singh and K. M. Krishna, "Reactive collision avoidance for multiple robots by non linear time scaling," in *IEEE Conf. on Decision and Control*, 2013.

[29] I. Ulrich and J. Borenstein, "Vfh*: local obstacle avoidance with look-ahead verification," in *IEEE Int. Conf. on Robotics and Automation*, 2000.

[30] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidnace," *Int J. of Robotics Research*, vol. 1, no. 14, 2015.

[31] J. van den Berg, J. Snape, S. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *IEEE Int. Conf. on Robotics and Automation*, 2011.

[32] M. Rufli, J. Alonso-Mora, and R. Siegwart, "Reciprocal collision avoidance with motion continuity constraints," *IEEE Trans. on Robotics*, vol. 29, no. 4, pp. 899–911, 2013.

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

[34] J. Israelsen, M. Beall, D. Bareiss, D. Stuart, E. Keeney, and J. van den Berg, "Automatic collision avoidance for manually tele-operated unmanned aerial vehicles," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 6638–6643.

[35] D. Bareiss, J. van den Berg, and K. K. Leang, "Stochastic automatic collision avoidance for tele-operated unmanned aerial vehicles," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015.

[36] D. Bareiss, J. Bourne, and K. K. Leang, "On-board model-based automatic collision avoidance: an application in remotely piloted unmanned aerial vehicles," *Autonomous Robots*, (Submitted) 2016.

[37] D. Bareiss, J. van den Berg, J. Abbott, and K. K. Leang, "Study of improved pilot performance using automatic collision avoidance for tele-operated unmanned aerial vehicles," in *IEEE Int. Symp. on Safety, Security, and Rescue Robots*, (To Be Submitted) 2016.

[38] B. Kluge and E. Prassler, "Reflective navigation: individual behaviors and group behaviors," in *IEEE Int. Conf. on Robotics and Automation*, 2004.

[39] J. van den Berg, M. Ling, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE Int. Conf. on Robotics and Automation*, 2008.

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

[41] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Proc. of the 10th International Symposium on Distributed Autonomous Robotic Systems*, 2010.

[42] J. Snape, J. van den Berg, S. Guy, and D. Manocha, "Smooth and collision-free navigation for multiple robots under differential-drive contstraints," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2010.

[43] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots," in *IEEE Int. Conf. on Robotics and Automation*, 2012.

[44] E. Lalish and K. Morgansen, "Distributed reactive collision avoidance," *Autonomous Robots*, vol. 32, no. 3, pp. 207–226, 2012.

[45] 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.

[46] 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.

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

[48] 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," *IEEE Trans. on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 392–405, 2008.

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

[50] 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.

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

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

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

[54] T. L. Gibo, L. N. Verner, D. D. Yuh, and A. M. Okamura, "Design considerations and human-machine performance of moving virtual fixtures," in *IEEE Int. Conf. on Robotics and Automation*, 2009.

CHAPTER 2

GENERALIZED RECIPROCAL COLLISION AVOIDANCE

The work in this chapter completes Objective 1 of the research and it was published in the *International Journal of Robotics Research* in 2015. The paper formalizes a generalized representation of reciprocal collision avoidance. Previously, reciprocal collision avoidance consisted of many approaches applied to specific equations of motion for a given robot system. In this paper, these approaches are unified through Control Obstacles. It is also shown how this approach can be extended to systems of robots with different, non-linear equations of motion, which was previously not possible. The approach is verified both in simulations and physical experiments. The reprint here is with permission.

D. Bareiss and J. van den Berg, "Generalized Reciprocal Collision Avoidance," in *International Journal of Robotics Research*, 34(12):1501-1514, Oct. 2015.

Article

The International Journal of Robotics Research
2015, Vol. 34(12) 1501–1514

DOI: 10.1177/0278364915576234
ijr.sagepub.com

Generalized reciprocal collision avoidance

Daman Bareiss[1] and Jur van den Berg[2]

Abstract

Reciprocal collision avoidance has become a popular area of research over recent years. Approaches have been developed for a variety of dynamic systems ranging from single integrators to car-like, differential-drive, and arbitrary, linear equations of motion. In this paper, we present two contributions. First, we provide a unification of these previous approaches under a single, generalized representation using control obstacles. In particular, we show how velocity obstacles, acceleration velocity obstacles, continuous control obstacles, and LQR-obstacles are special instances of our generalized framework. Secondly, we present an extension of control obstacles to general reciprocal collision avoidance for non-linear, non-homogeneous systems where the robots may have different state spaces and different non-linear equations of motion from one another Previous approaches to reciprocal collision avoidance could not be applied to such systems, as they use a relative formulation of the equations of motion and can, therefore, only apply to homogeneous, linear systems where all robots have the same linear equations of motion. Our approach allows for general mobile robots to independently select new control inputs while avoiding collisions with each other We implemented our approach in simulation for a variety of mobile robots with non-linear equations of motion: differential-drive, differential-drive with a trailer, car-like, and hovercrafts. We also performed physical experiments with a combination of differential-drive, differential-drive with a trailer, and car-like robots. Our results show that our approach is capable of letting a non-homogeneous group of robots with non-linear equations of motion safely avoid collisions at real-time computation rates.

Keywords

Collision avoidance, multi-robot system, decentralized control, mobile robot navigation, motion control

1. Introduction

Collision avoidance is a fundamental problem in robotics. The problem can generally be defined in the context of an autonomous mobile robot navigating in an environment with obstacles and/or other moving entities, where the robot employs a continuous sensing-control cycle. In each cycle, the robot must compute an action based on its local observations of the environment, such that it stays free of collisions with the moving obstacles and the other robots, and progresses towards a goal. Many works in robotics have addressed the problem of collision avoidance with moving obstacles (Fox et al., 1997; Fiorini and Shiller, 1998; Hsu et al., 2002; Petti and Fraichard, 2005). Typically, these approaches predict where the moving obstacles might be in the future by extrapolating their observed trajectories, and let the robot avoid collisions accordingly. Velocity obstacles (VO) (Fiorini and Shiller, 1998) formalize this principle by characterizing the set of velocities for the robot that result in a collision at some future time. Continually selecting a velocity outside of this set will then guarantee collision-free navigation for the robot.

However, such approaches are insufficient when the robot encounters other robots that also actively make decisions based on their surroundings: considering them as moving obstacles overlooks the fact that they react to the robot in the same way the robot reacts to them, and inherently causes suboptimal and oscillatory motion (Kluge and Prassler, 2004; Van den Berg et al., 2008).

This has lead to the development of *reciprocal collision avoidance* techniques, which specifically account for the reactive nature of the other robots without relying on coordination or communication among robots. The earliest approaches were direct extensions of VO, in which each robot is given half the responsibility of avoiding pairwise collisions (Van den Berg et al., 2008, 2009). Since this

[1]Department of Mechanical Engineering, University of Utah, UT, USA
[2]School of Computing, University of Utah, UT, USA

Corresponding author:
Daman Bareiss, Department of Mechanical Engineering, University of Utah, 50 S. Central Campus Dr., 2110 MEB, Salt Lake City, UT 841412, USA.
Email address: daman.bareiss@utah.edu

Table 1. Classification of reciprocal collision avoidance approaches.

	Homogeneous	Non-homogeneous
Linear	VO/ORCA AVO CCO LQR–obstacles	**Control obstacles**
Non-linear	**Control obstacles** **Control obstacles**	**Control obstacles**

approach only applies to robots with very simple dynamics that allow the robots to change their velocity instantaneously, most subsequent research on the topic has focused on extending the approach to robots with more complex dynamic constraints, such as differential-drive (Alonso-Mora et al., 2010; Snape et al., 2010), car-like (Alonso-Mora et al., 2012), double-integrator (Lalish and Morgansen, 2012; Van den Berg et al., 2012), arbitrary integrator (Ruffli et al., 2013), and robots with linear quadratic regulator (LQR) controllers (Bareiss and van den Berg, 2013). A major limitation of these approaches though is that all robots are assumed to have exactly the same equations of motion, in other words, they apply to *homogeneous* systems only. Moreover, in all these approaches the assumed equations of motion are *linear*, as these approaches rely on the ability to express the *relative* motion of pairs of robots in terms of the *relative* control input (i.e. the difference between the control inputs) of the robots. Hence, they do not apply to non-homogeneous or non-linear systems, where robots have different and/or non-linear equations of motion, which limits their applicability to real-world robots and in real-world applications.

In this paper, we address this shortcoming by presenting a new reciprocal collision avoidance method with two main contributions (see Table 1):

- First, we provide a unification of all previous approaches to reciprocal collision avoidance under a single, generalized representation using *control obstacles*. We will show specifically that approaches such as VO (Fiorini and Shiller, 1998), acceleration velocity obstacles (AVO) (Van den Berg et al., 2012), continuous control obstacles (CCO) (Ruffli et al., 2013), and LQR-obstacles (Bareiss and van den Berg, 2013) are each a special instance of our generalized framework. Moreover, we will show that our formulation is generally applicable to all homogeneous systems with linear equations of motion, and as such covers that entire class of systems.
- Second, we present an extension of control obstacles to reciprocal collision avoidance for general non-linear and/or non-homogeneous systems where the robots may have different state spaces and different non-linear equations of motion. No previous approaches to reciprocal collision avoidance could be applied to these categories of systems, even though some previous work has shown how specific instances of non-linear systems can be turned into a linear system formulation to which one of the previous approaches could be applied (see the next section for a more thorough discussion).

We implemented our approach in simulation for a variety of mobile robots with non-linear equations of motion: differential-drive, differential-drive with a trailer, car-like, and hovercrafts. We also performed physical experiments with a combination of differential-drive, differential-drive with a trailer, and car-like robots. Our results show that our approach is capable of letting a non-homogeneous group of robots with non-linear equations of motion safely avoid collisions at real-time computation rates.

The remainder of the paper is structured as follows. Section 2 reviews previous approaches to reciprocal collision avoidance. Section 3 formally defines the problem of reciprocal collision avoidance. Section 4 presents our generalized approach for homogeneous systems with linear equations of motion using control obstacles. Section 5 shows how the previous reciprocal collision avoidance approaches can be represented in our generalized approach. In Section 6 we explore the potential of our approach to non-homogeneous systems with non-linear equations of motion. Section 7 presents our results and Section 8 summarizes and concludes.

2. Previous work

One of the early developments in collision avoidance was the velocity obstacle (VO) (Fiorini and Shiller, 1998). The VO is defined as a cone in the velocity space based on relative positions and geometries which defines all relative velocities which will result in a collision. To avoid a collision, the robot needs to select a new velocity that lies outside the VO.

The approach of the VO was initially developed for a single active agent avoiding collisions with passive agents or moving obstacles. The approach was extended to perform reciprocal collision avoidance between two active agents in reciprocal velocity obstacles (RVO) (Van den Berg et al., 2008). In RVO, the concept of the VO is used but each robot must take half the responsibility to avoid collisions rather than the entire responsibility as in the VO algorithm. However, as the number of agents increases RVO tends to result in oscillatory motions. Optimal reciprocal velocity obstacles (ORCA) (Van den Berg et al., 2009) was developed to address this issue. In ORCA, the set of safe velocities is evenly divided between two robots by defining halfplanes of safe, possible velocities. These halfplanes are defined with respect to the VO and are the sets of individual velocities for two robots that result in *relative* velocities outside of the VO, thus avoiding collisions. Each robot selects a new velocity from the set of

safe, possible velocities that is as close as possible to a target velocity.

These algorithms all have the limitation that they are only guaranteed to provide safe, collision-free motion for robots with the linear equation of motion $\dot{\mathbf{p}} = \mathbf{v}$ where the position $\mathbf{p}$ defines the state and the velocity $\mathbf{v}$ defines the control input. This model is not practical for most real-world robots. Several extensions of these earlier approaches were introduced to address this issue and present reciprocal collision avoidance for more complicated dynamics. AVO (Van den Berg et al., 2012) were defined for robots with states consisting of position and velocity $\mathbf{x} = [\mathbf{p}^{\mathrm{T}}\mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$ with acceleration control inputs $\mathbf{u} = k(\mathbf{v}^{\star} - \mathbf{v})$ where k is some proportional gain, $\mathbf{v}^{\star}$ is some target velocity, and $\mathbf{v}$ is the current velocity. The approach in AVO was further generalized to apply to arbitrary-degree integrators through CCO by Ruffli et al. (2013). Further extension was provided by Bareiss and van den Berg (2013) for robots with arbitrary, homogeneous, linear equations of motion.

Table 1 summarizes the types of multi-robot system that have been considered in reciprocal collision avoidance based on a high-level categorization along two axes:

- *Homogeneous versus non-homogeneous systems:* *Homogeneous* teams of robots consist of robots that all have exactly the same state space and equations of motion, for example all robots are single-integrators with directly controllable velocity. *Non-homogeneous* teams of robots on the other hand may include robots with different state spaces and equations of motion, for example a single-integrator interacting with a double-integrator (a robot with directly controllable acceleration).
- *Linear versus non-linear equations of motion*: For systems with *linear* equations of motion, the derivative of the state is a linear function of the state and the control input, as is the case with single and double integrators for example. Differential-drive and car-like robots are examples of systems with *non-linear* equations of motion.

It turns out that all existing approaches to reciprocal collision avoidance are limited to *specific* instances of homogeneous systems with linear equations of motion, as these approaches are reliant upon the ability to express the equations of motion in terms of their current *relative* states and *relative* control inputs, which is generally not possible for non-homogeneous and/or non-linear systems. Our approach extends this previous work and is generally applicable across this two-dimensional spectrum.

There have been some developments for reciprocal collision avoidance for non-linear equations of motion. Reciprocal collision avoidance for differential-drive robots was performed by Snape et al. (2010) where the center of the robots was shifted and the bounding radius was increased in order to model the robots as holonomic with linear equations of motion. In Alonso-Mora et al. (2010), non-holonomic ORCA (NH-ORCA) was developed. NH-ORCA increases the radius of the robot based on the error in a tracking controller which allows non-holonomic robots to track holonomic trajectories as demonstrated for differential-drive robots. The NH-ORCA algorithm is applied to car-like robots by Alonso-Mora et al. (2012). In Van den Berg et al. (2012) it was shown how car-like robots can be represented as double integrators, to which AVO can be applied. These approaches all have in common that they transform specific instances of non-linear equations of motion into a linear formulation to which reciprocal collision avoidance can be applied. Our approach, in contrast, will apply directly to any general non-linear equations of motion.

Our work has some similarities to non-linear velocity obstacles (NLVO) (Shiller et al., 2001) and generalized velocity obstacles (GVO) (Wilkie et al., 2009). The NLVO algorithm expands the VO algorithm to allow for a robot with linear equations of motion to avoid collisions with passive obstacles moving with known, possibly non-linear trajectories. The self-motion velocity obstacle (SMVO) is another approach that utilizes the NLVO while considering more general robot trajectories (Shiller et al., 2008). The GVO algorithm does basically the opposite of the NLVO by defining a "control obstacle" for robots with non-linear equations of motion to avoid a passive obstacle moving along a linear trajectory. This approach samples the space of possible control inputs to determine if a collision will occur in the future. Neither of these approaches can be trivially extended to reciprocal collision avoidance.

3. Problem statement

3.1. Notation

We use the following notational conventions in this paper. Vector sets $\mathcal{A}$ are denoted using calligraphics, vectors $\mathbf{a}$ are denoted using boldface, matrices A are denoted using upper-case italics, and scalars a are denoted by lower-case italics. Scalar and matrix multiplication, and Minkowski sums of sets, are defined as

$$a\mathcal{X} = \{a\mathbf{x} | \mathbf{x} \in \mathcal{X}\}, \quad A\mathcal{X} = \{A\mathbf{x} | \mathbf{x} \in \mathcal{X}\}$$
$$\mathcal{X} \oplus \mathcal{Y} = \{\mathbf{x} + \mathbf{y} | \mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}\}$$

It follows that $\mathcal{A} \oplus \{\mathbf{a}\}$ denotes a translation of a set $\mathcal{A}$ by a vector $\mathbf{a}$.

3.2. Problem setup

We consider multiple mobile robots sharing a common workspace where the robots have potentially different, non-linear equations of motion and state spaces. Let the state

space of robot i be $\mathcal{X}_i \subset \mathbb{R}^{n_i}$. Let $\mathbb{R}^d$ be the robots' physical workspace, where $d = 2$ or $d = 3$ typically. We assume the position $\mathbf{p}_i \in \mathbb{R}^d$ of robot i can be derived from its state $\mathbf{x}_i \in \mathcal{X}_i$ by some potentially non-linear projection function $\mathbf{q}_i \in \mathcal{X}_i \to \mathbb{R}^d$:

$$\mathbf{p}_i(t) = \mathbf{q}_i(\mathbf{x}_i(t)) \tag{1}$$

Let $\mathcal{O}_i \subset \mathbb{R}^d$ be the geometry of robot i relative to its position. We assume that the geometric shape of the robot is determined only by its position and not its orientation, in other words, it is rotationally invariant. More specifically, we consider the robot geometry as its bounding circle similar to the approach in the original VO (Fiorini and Shiller, 1998). This is a reasonable assumption for most mobile robots that greatly simplifies the development of our approach. See Giese et al. (2014) for work specifically including the orientation dimension in reciprocal collision avoidance.

We further assume that the dimension of the control input is equal to the dimension of the workspace, where $\mathcal{U}_i \subset \mathbb{R}^d$ is the valid control input space, which is assumed to be convex. Let the continuous-time equation of motion for robot i be given by a potentially non-linear function $\mathbf{f}_i \in \mathcal{X}_i \times \mathcal{U}_i \to \mathbb{R}^{n_i}$:

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

where $\mathbf{x}_i(t)$ is the state and $\mathbf{u}_i(t)$ is the control input at time t for robot i. It is important to remember that $\mathcal{X}_i$, $\mathbf{q}_i$, and $\mathbf{f}_i$ may be different for every robot i.

Given a current state $\mathbf{x}_i = \mathbf{x}_i(0)$ of robot i and some constant control input $\mathbf{u}_i = \mathbf{u}_i(0)$, the state of the robot at a given time $t > 0$ is given by

$$\mathbf{x}_i(t) = \mathbf{g}_i(t, \mathbf{x}_i, \mathbf{u}_i) \tag{3}$$

where $\mathbf{g}_i \in \mathbb{R} \times \mathcal{X}_i \times \mathcal{U}_i \to \mathcal{X}_i$ is the solution to the differential equation of equation (2), which can be obtained numerically, for example through a Runge–Kutta integration.

3.3. Problem statement

The problem of reciprocal collision avoidance we are addressing can now be defined as having each robot i independently compute a change $\Delta\mathbf{u}_i \in \mathcal{U}_i \oplus \{-\mathbf{u}_i\}$ of its current control input $\mathbf{u}_i$ given the current states $\mathbf{x}_j$ and control inputs $\mathbf{u}_j$ of all other robots $j \neq i$, such that the robots do not collide within a time horizon τ:

$$\begin{aligned}\forall(j \neq i, 0 \leq t < \tau) :: (\mathcal{O}_i \oplus \{\mathbf{q}_i(\mathbf{g}_i(t, \mathbf{x}_i, \mathbf{u}_i + \Delta\mathbf{u}_i))\}) \cap \\ (\mathcal{O}_j \oplus \{\mathbf{q}_j(\mathbf{g}_j(t, \mathbf{x}_j, \mathbf{u}_j + \Delta\mathbf{u}_j))\}) = \emptyset\end{aligned} \tag{4}$$

3.4. Challenges and assumptions

The challenge of reciprocal collision avoidance is that robot i does not know the change in control input $\Delta\mathbf{u}_j$ the other robots are going to choose. Therefore, we rely on the assumption that all robots use the same algorithm in order to select their change of control input. In this paper we discuss the design of an algorithm to compute changes in control inputs such that collision avoidance is achieved. In doing so, we make the following assumptions:

(I) When computing a control input, we assume that it remains constant over finite time τ into the future. The actual sensing-action cycle is much shorter than τ and a new control input is computed in every sensing-action cycle.

(II) We assume that the robots can fully observe each other's state and control input.

(III) We assume that the robots have the same type of control input, for example desired velocity, which is equal in dimension to the dimension of the workspace.

4. Generalized reciprocal collision avoidance for homogeneous, linear equations of motion

In this section we introduce the general concept of *control obstacles* that applies to general *linear* and *homogeneous* systems. The control obstacle generalizes all previous approaches on reciprocal collision avoidance, as we will show in Section 5. In Section 6 we present the extension to non-linear, non-homogeneous systems.

4.1. Control obstacles

In this section, we consider a system of robots that all have the same linear equations of motion, that is, a linear, homogeneous system. Equation (1) can be expressed for all robots i in a linear, homogeneous system as

$$\mathbf{p}_i(t) = \mathbf{q}(\mathbf{x}_i(t)) = C\mathbf{x}_i(t) + \mathbf{d} \tag{5}$$

where the matrix $C \in \mathbb{R}^{d \times n}$ and the vector $\mathbf{d} \in \mathbb{R}^d$ map a robot's state to its position and are identical for all robots.

Let the state space $\mathcal{X}_i = \mathcal{X} \subset \mathbb{R}^n$ be identical for all robots i. Equation (2) can be expressed for all robots i as

$$\dot{\mathbf{x}}_i(t) = \mathbf{f}(\mathbf{x}_i(t), \mathbf{u}_i(t)) = A\mathbf{x}_i(t) + B\mathbf{u}_i(t) + \mathbf{c} \tag{6}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times d}$, and $\mathbf{c} \in \mathbb{R}^n$.

Given a current state $\mathbf{x}_i = \mathbf{x}_i(0)$ and a constant control input $\mathbf{u}_i = \mathbf{u}_i(0)$, solving the differential equation in equation (6) gives

$$\mathbf{x}_i(t) = \mathbf{g}(t, \mathbf{x}_i, \mathbf{u}_i) = F(t)\mathbf{x}_i + G(t)\mathbf{u}_i + \mathbf{h}(t) \tag{7}$$

where $F(t) \in \mathbb{R} \to \mathbb{R}^{n \times n}$, $G(t) \in \mathbb{R} \to \mathbb{R}^{n \times d}$, and $\mathbf{h}(t) \in \mathbb{R} \to \mathbb{R}^n$ are identical for every robot and are given as

$$\begin{bmatrix} F(t) & G(t) & \mathbf{h}(t) \\ 0 & I & 0 \\ 0 & 0 & 1 \end{bmatrix} = \exp\left(t \begin{bmatrix} A & B & \mathbf{c} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right) \tag{8}$$

Remember that for homogeneous systems with linear equations of motion, $\mathbf{q}$, $\mathbf{f}$, and $\mathbf{g}$ are the same for all robots. A unique property for these systems is that

$$\begin{aligned}\mathbf{g}(t, \mathbf{x}_i, \mathbf{u}_i) - \mathbf{g}(t, \mathbf{x}_j, \mathbf{u}_j) &= \mathbf{g}(t, \mathbf{x}_i - \mathbf{x}_j, \mathbf{u}_i - \mathbf{u}_j) \\ &= \mathbf{g}(t, \mathbf{x}_{ij}, \mathbf{u}_{ij}) \\ &= F(t)\mathbf{x}_{ij} + G(t)\mathbf{u}_{ij} + \mathbf{h}(t)\end{aligned} \tag{9}$$

where $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$ are the relative initial states and relative control inputs, respectively. This property has been exploited by all previous work on reciprocal collision avoidance. However, it does not hold for homogeneous systems with non-linear equations of motion, non-homogeneous systems with linear equations of motion, or non-homogeneous systems with non-linear equations of motion. We will discuss these cases in Section 6.

Substituting equation (7) into equation (5) for a given control input $\mathbf{u}_i + \Delta\mathbf{u}_i$, where $\mathbf{u}_i$ is the current control input and $\Delta\mathbf{u}_i$ is the change in control input, gives

$$\mathbf{p}_i(t) = \hat{\mathbf{p}}(t, \mathbf{x}_i, \mathbf{u}_i) + J(t)\Delta\mathbf{u}_i \tag{10}$$

where $\hat{\mathbf{p}}(t, \mathbf{x}_i, \mathbf{u}_i) \in \mathbb{R}^d$ and $J(t) \in \mathbb{R}^{d \times d}$ are

$$\hat{\mathbf{p}}(t, \mathbf{x}_i, \mathbf{u}_i) = C(F(t)\mathbf{x}_i + G(t)\mathbf{u}_i + \mathbf{h}(t)) + \mathbf{d} \tag{11}$$

$$J(t) = CG(t) \tag{12}$$

Given equation (10), we now define the control obstacle $\mathcal{UO}_{ij}$ for a robot i avoiding collisions with robot j. In order for the robots to avoid collision, their relative position $\mathbf{p}_{ij}(t) = \mathbf{p}_i(t) - \mathbf{p}_j(t)$ must remain outside of the Minkowski sum of the robots' geometries $\mathcal{O}_{ij} = \mathcal{O}_j \oplus -\mathcal{O}_i$:

$$\mathbf{p}_{ij}(t) \notin \mathcal{O}_{ij} \tag{13}$$

Substituting equation (10) into equation (13) and solving for the relative change in control input $\Delta\mathbf{u}_{ij}$ gives

$$\begin{aligned}&\hat{\mathbf{p}}(t, \mathbf{x}_{ij}, \mathbf{u}_{ij}) + J(t)\Delta\mathbf{u}_{ij} \notin \mathcal{O}_{ij} \Rightarrow \Delta\mathbf{u}_{ij} \notin J(t)^{-1} \\ &\quad (\mathcal{O}_{ij} \oplus \{-\hat{\mathbf{p}}(t, \mathbf{x}_{ij}, \mathbf{u}_{ij})\})\end{aligned} \tag{14}$$

Equation (14) represents a constraint on the change in relative control input $\Delta\mathbf{u}_{ij}$ such that robots i and j do not collide at time t. We define the control obstacle as the union of equation (14) for all time t less than the time horizon τ:

$$\mathcal{UO}_{ij} = \bigcup_{0 \leq t < \tau} J(t)^{-1}(\mathcal{O}_{ij} \oplus \{-\hat{\mathbf{p}}(t, \mathbf{x}_{ij}, \mathbf{u}_{ij})\}) \tag{15}$$

In other words, a collision will not occur between robot i and robot j within τ time into the future when their relative change in control input $\Delta\mathbf{u}_{ij}$ lies outside the control obstacle:

$$\Delta\mathbf{u}_{ij} \notin \mathcal{UO}_{ij} \tag{16}$$

The geometry of $\mathcal{UO}_{ij}$ can be seen as a union of copies of the relative geometry $\mathcal{O}_{ij}$, each translated to $-\hat{\mathbf{p}}(t, \mathbf{x}_{ij}, \mathbf{u}_{ij})$, that is, the nominal trajectory of robot j relative to robot i, and then transformed by J^{-1}. If the geometries of the robots are discs, $\mathcal{UO}_{ij}$ is hence a union of ellipsoids.

4.2. Avoiding collisions with passive robots

For a passive robot or environmental object, we can assume that $\Delta\mathbf{u}_j = \mathbf{0}$. That is, we assume the other robot does not change its control input. Avoiding collisions with that robot or object can then be performed simply by selecting a change in control input $\Delta\mathbf{u}_i$ outside the control obstacle:

$$\Delta\mathbf{u}_i \notin \mathcal{UO}_{ij} \tag{17}$$

For the case where it cannot be assumed that $\Delta\mathbf{u}_j = \mathbf{0}$, in other words, both robots are actively avoiding collisions, reciprocal collision avoidance must be performed, which we discuss next.

4.3. Reciprocal collision avoidance using control obstacles

Equation (16) gives the constraint on the relative change in control input $\Delta\mathbf{u}_{ij}$ for two robots to avoid collisions. When it cannot be assumed that $\Delta\mathbf{u}_j = \mathbf{0}$, robot i has to consider the change in control input $\Delta\mathbf{u}_j$ robot j is going to select in order for robot i to select a safe change in control input $\Delta\mathbf{u}_i$ for itself. The challenge is that $\Delta\mathbf{u}_j$ is unknown to robot i and the robots are not allowed to communicate. Hence, our approach is that robot i computes sets $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ of possible *safe* changes of control inputs for robot i and robot j, respectively, that satisfy the constraint

$$((\mathcal{RCA}_{ij} \cap \tilde{\mathcal{U}}_i) \oplus -(\mathcal{RCA}_{ji} \cap \tilde{\mathcal{U}}_j)) \cap \mathcal{UO}_{ij} = \emptyset \tag{18}$$

where $\tilde{\mathcal{U}}_i = \mathcal{U}_i \oplus \{-\mathbf{u}_i\}$ is the set of feasible changes in control input for robot i given the control input constraints. If robot i selects a change in control input $\Delta\mathbf{u}_i$ from $\mathcal{RCA}_{ij}$ and robot j selects a change in control input $\Delta\mathbf{u}_j$ from $\mathcal{RCA}_{ji}$, which each satisfy their respective control input constraints, then it is guaranteed that $\Delta\mathbf{u}_{ij} \notin \mathcal{UO}_{ij}$ and the robots will not collide within τ time in the future. We will let robot i compute $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ in such a way that if robot j were to apply the same algorithm to its situation, it would compute the same sets $\mathcal{RCA}_{ji}$ and $\mathcal{RCA}_{ij}$. Robot i is then free to choose any change in control input from the set $\mathcal{RCA}_{ij}$ to avoid collisions with robot j.

There are infinitely many pairs of sets of changes in control inputs $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ that satisfy equation (18). Therefore, we choose to find a pair of sets that divides the responsibility of avoiding collisions equally between both robots. Let us define a convex set $\mathcal{C}$ of safe *relative* changes in control inputs such that

$$(\mathcal{C} \cap \tilde{\mathcal{U}}_{ij}) \cap \mathcal{UO}_{ij} = \emptyset \tag{19}$$

where $\tilde{\mathcal{U}}_{ij} = \tilde{\mathcal{U}}_i \oplus -\tilde{\mathcal{U}}_j$ represents the feasible relative changes in control inputs. Any relative change in control

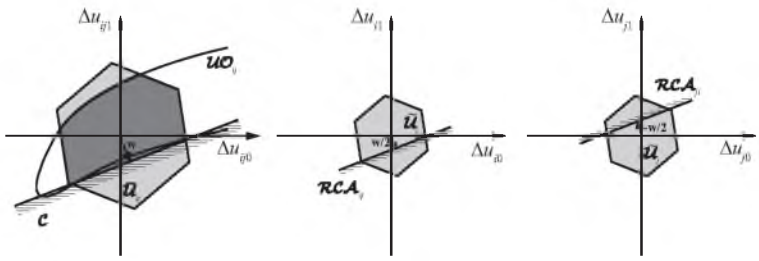

Fig. 1. On the left, a control obstacle $\mathcal{UO}_{ij}$ is given by its outline. The set of feasible relative changes in control input $\tilde{\mathcal{U}}_{ij}$ (i.e. those that adhere to the control input constraints) is shown as the light gray hexagon. The minimum change in relative control input required to avoid collision is shown as the vector $\mathbf{w}$ which defines the position of the halfspace $\mathcal{C}$. The vector $\mathbf{w}$ is defined as the closest point to the origin outside the *convex hull* (dark gray region) of the intersection of the control obstacle and the feasible changes in control input $\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij}$. Each robot constructs a set of safe changes in control input, $\mathcal{RCA}_{ij}$ for robot i and $\mathcal{RCA}_{ji}$ for robot j, at $\mathbf{w}/2$ and $-\mathbf{w}/2$ respectively as shown in the middle and right images.

input $\Delta\mathbf{u}_{ij}$ that does not violate control input constraints and that is within $\mathcal{C}$, in other words any $\Delta\mathbf{u}_{ij} \in (\mathcal{C} \cap \tilde{\mathcal{U}}_{ij})$, will avoid collisions between robots i and j within τ time.

Since for convex sets it holds that $\mathcal{X} = \frac{1}{2}\mathcal{X} \oplus \frac{1}{2}\mathcal{X}$, the set $\mathcal{C}$ can be "halved" to determine sets of safe changes in control input for both robot i and robot j as

$$\mathcal{RCA}_{ij} = \frac{1}{2}\mathcal{C}, \qquad \mathcal{RCA}_{ji} = -\frac{1}{2}\mathcal{C} \tag{20}$$

where the desire to divide the responsibility for collision avoidance equally between the two robots motivates halving of the set $\mathcal{C}$ for each robot. We will define the convex set $\mathcal{C}$ more concretely below, but if it satisfies the condition of equation (19), we can prove that our definition of $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ in equation (20) satisfies our constraint on $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ of equation (18):

$$\begin{aligned}
&((\mathcal{RCA}_{ij} \cap \tilde{\mathcal{U}}_i) \oplus -(\mathcal{RCA}_{ji} \cap \tilde{\mathcal{U}}_j)) \cap \mathcal{UO}_{ij} \\
&= \left(\left(\frac{1}{2}\mathcal{C} \cap \tilde{\mathcal{U}}_i\right) \oplus -\left(-\frac{1}{2}\mathcal{C} \cap \tilde{\mathcal{U}}_j\right)\right) \cap \mathcal{UO}_{ij} \\
&\subseteq \left(\left(\frac{1}{2}\mathcal{C} \oplus \frac{1}{2}\mathcal{C}\right) \cap (\tilde{\mathcal{U}}_i \oplus -\tilde{\mathcal{U}}_j)\right) \cap \mathcal{UO}_{ij} \\
&= (\mathcal{C} \cap \tilde{\mathcal{U}}_{ij}) \cap \mathcal{UO}_{ij} \\
&= \emptyset
\end{aligned} \tag{21}$$

where we use the fact that

$$\begin{aligned}
(\mathcal{W} \cap \mathcal{X}) \oplus (\mathcal{Y} \cap \mathcal{Z}) = (\mathcal{W} \oplus \mathcal{Y}) \cap (\mathcal{W} \oplus \mathcal{Z}) \cap \\
(\mathcal{X} \oplus \mathcal{Y}) \cap (\mathcal{X} \oplus \mathcal{Z}) \\
\subseteq (\mathcal{W} \oplus \mathcal{Y}) \cap (\mathcal{X} \oplus \mathcal{Z})
\end{aligned}$$

What remains is choosing the convex set $\mathcal{C}$ of safe relative changes in control inputs. Ideally $\mathcal{C}$ should be the largest set of safe relative changes in control input, but such a set can be difficult to compute exactly. Therefore, we define $\mathcal{C}$ to be the halfspace tangent to the convex hull of the set of feasible relative control inputs that will result in a collision, that is, $\mathcal{CH}(\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij})$, at the point $\mathbf{w}$ on the convex hull's boundary closest to the origin:

$$\mathbf{w} = \underset{\mathbf{u} \in \partial\mathcal{CH}(\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij})}{\operatorname{argmin}} \|\mathbf{u}\| \tag{22}$$

$$\mathcal{C} = \begin{cases} \{\mathbf{u} | (\mathbf{u} - \mathbf{w}) \cdot \mathbf{w} \geq \mathbf{0}\} & \text{if } \mathbf{0} \in \mathcal{CH}(\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij}) \\ \{\mathbf{u} | (\mathbf{u} - \mathbf{w}) \cdot \mathbf{w} \leq \mathbf{0}\} & \text{if } \mathbf{0} \notin \mathcal{CH}(\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij}) \end{cases} \tag{23}$$

where ∂ refers to the boundary of a set. By construction, this definition of $\mathcal{C}$ satisfies equation (19).

This is illustrated in Figure 1. The set of feasible relative changes in control input $\tilde{\mathcal{U}}_{ij}$ (i.e. those that adhere to control input constraints) is represented by the light gray hexagon. The dark gray region represents the convex hull of the intersection of the feasible relative changes in control input and the control obstacle, that is, $\mathcal{CH}(\mathcal{UO}_{ij} \cap \tilde{\mathcal{U}}_{ij})$. The set $\mathcal{C}$ is shown located tangent to this convex hull at the point closest to the origin. Placing the set $\mathcal{C}$ at the closest point to the origin represents the desire to keep the relative changes in control input as small as possible and, therefore, allow the robots to maintain their current, desired control input as closely as possible.

Since $\mathcal{C}$ is a halfspace, it follows that $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ are halfspaces as well. If the robots are currently on a collision course, that is, $\Delta\mathbf{u}_{ij} \in \mathcal{UO}_{ij}$, the vector $\mathbf{w}$ represents the smallest relative change in control input required to avoid a collision. Given that the two robots share the responsibility for avoiding collisions equally, the sets $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ are halfspaces located at $\frac{1}{2}\mathbf{w}$ and $-\frac{1}{2}\mathbf{w}$ from the origin of their respective control input spaces, as shown in Figure 1.

It is important to note that each robot i and j can independently compute their halfspaces $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$ since the construction of the control obstacle from robot j's perspective $\mathcal{UO}_{ji}$ results in the same sets $\mathcal{RCA}_{ji}$ and $\mathcal{RCA}_{ij}$ since $\mathcal{UO}_{ij} = -\mathcal{UO}_{ji}$.

If both robots desire to keep their changes in control inputs as small as possible while ensuring they avoid collisions, each robot selects a change in control input as

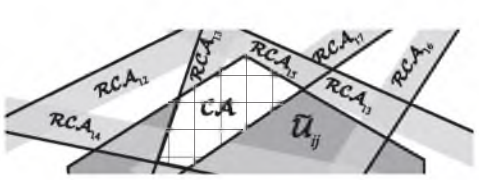

Fig. 2. A scenario for a group of seven robots avoiding collision. Robot 1 creates a safe set of changes in control inputs $\mathcal{RCA}_{1j}$ for every other robot j. The intersection of the union of these planes and the space of possible changes of control input $\tilde{\mathcal{U}}_{ij}$ is shown as $\mathcal{CA}$, which is the set of changes in control input that avoid collisions with every other robot while adhering to control input constraints.

$$\Delta\mathbf{u}_i^{\min} = \underset{\Delta\mathbf{u}_i \in \mathcal{RCA}_{ij}}{\operatorname{argmin}} \|\Delta\mathbf{u}_i\| \tag{24}$$

Given the symmetry of $\mathcal{RCA}_{ij}$ and $\mathcal{RCA}_{ji}$, it follows that

$$\Delta\mathbf{u}_j^{\min} = -\Delta\mathbf{u}_i^{\min} \tag{25}$$

We use this observation later in the extension of the reciprocal collision avoidance approach to non-linear systems.

4.4. Avoiding collisions with multiple robots

A control obstacle is defined for pairwise collision avoidance but can easily be extended to more than two robots with an approach similar to Van den Berg et al. (2009). Each robot i creates a control obstacle and determines its set of safe potential changes in control input $\mathcal{RCA}_{ij}$ with respect to *every* other robot j, as in Figure 2. After considering every robot j, the change in control input for robot i is selected that is safe from all collisions:

$$\Delta\mathbf{u}_i \in \tilde{\mathcal{U}}_i \cap \bigcap_{j \neq i} \mathcal{RCA}_{ij} \tag{26}$$

where $\Delta\mathbf{u}_i$ can be found with a convex optimization method in the d-dimensional space of control inputs similar to the method by Van den Berg et al. (2009).

Given a preferred change in control input, the convex optimization will result in a change in control input that is as close as possible to some preferred input while not violating equation (26). However, there can be cases in which the set of safe changes in control input may be empty, that is, $\tilde{\mathcal{U}}_i \cap \bigcap_{j \neq i} \mathcal{RCA}_{ij} = \emptyset$. When this occurs, a convex optimization in a $(d + 1)$-dimensional space will select the change of control input that will *least* violate the constraints. See Van den Berg et al. (2009) for full details. This potentially results in a collision within τ time if the control inputs truly remain constant, but given that a new control input is selected in each sensing-action cycle, in practice this turns out to typically result in safe motion. However, the fact that collision avoidance can only be theoretically guaranteed in some cases remains a limitation of our approach.

5. Generalization of previous reciprocal collision avoidance approaches

Above, we have developed a method for reciprocal collision avoidance for a homogeneous system of multiple robots with general, linear equations of motion. We did so through a new method of control obstacles. Previous approaches of reciprocal collision avoidance can be shown to be special cases of control obstacles. As shown in the previous sections, the control obstacle is fully defined for a system if given A, B, and $\mathbf{c}$ from equation (6) and C and $\mathbf{d}$ from equation (5). We will now show how previous methods of reciprocal collision avoidance can be represented as control obstacles using these terms.

5.1. VO

The VO algorithm assumes the robot's equations of motion are a single integrator kinematic model:

$$\dot{\mathbf{p}} = \mathbf{v} \tag{27}$$

where $\mathbf{x} = \mathbf{p}$ and $\mathcal{X} \subset \mathbb{R}^2$ is the space of positions, $\mathbf{u} = \mathbf{v}$ and $\mathcal{U} \subset \mathbb{R}^2$ is the space of velocities. For equation 27 we find

$$A = 0, \qquad B = I, \qquad \mathbf{c} = \mathbf{0}, \qquad C = I, \qquad \mathbf{d} = \mathbf{0}$$

Solving equation (7) for these, we find

$$F(t) = I, \qquad G(t) = tI, \qquad \mathbf{h}(t) = \mathbf{0}$$

When $\mathcal{O}_{ij}$ is a circle or sphere, the control obstacle is equivalent to the VO translated by the negative of the current relative input $-\mathbf{v}_{ij}$ as in Figure 3. This discrepancy arises from control obstacles being developed based on *changes* in control input rather than the absolute control input.

5.2. AVO

The AVO algorithm is presented by Van den Berg et al. (2012) as an alternative to the VO. One of the major problems with the VO is the assumption that instantaneous changes in velocity are possible. However, as this is not the case for physical systems, the AVO algorithm was developed.

In AVO, the robots have four state variables (the two-dimensional position and velocity) $\mathbf{x} \in \mathbb{R}^4$ and two control inputs (the two-dimensional acceleration) $\mathbf{u} \in \mathbb{R}^2$. The control inputs are driven by a proportional controller:

$$\dot{\mathbf{v}} = \frac{1}{\delta}(\mathbf{v}^{\star} - \mathbf{v}) \tag{28}$$

where $\mathbf{v}^{\star}$ is the desired velocity, $\mathbf{v}$ is the current velocity, and δ is a controller parameter.

Integrating equation (28) twice to obtain the system's trajectory gives

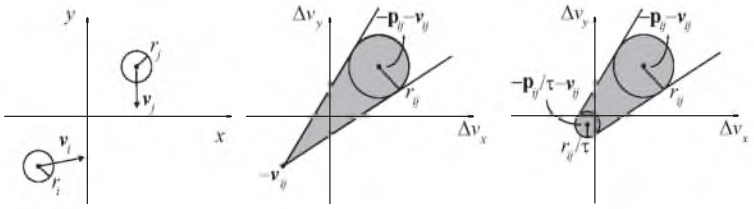

Fig. 3. Left: a pair of robots with equation of motion $\dot{\mathbf{p}} = \mathbf{v}$ where their current velocities will lead to a collision course with each other. Middle: an infinite time horizon control obstacle is given for the robot configuration on the left. As can be seen, the control obstacle contains the origin, meaning that the robots will indeed collide if they continue with their current control input. Right: the same control obstacle is shown, except now it is bounded by a finite time horizon τ. This control obstacle is equivalent to the VO for the single-integrator dynamics except it is shifted by the negative of the current relative velocity $-\mathbf{v}_{ij}$. This discrepancy between the control obstacle and VO arises because the control obstacle is defined in terms of the change in velocity rather than the absolute velocity.

$$\mathbf{v}(t) = \mathbf{v}^{\star} - e^{-t/\delta}\left(\mathbf{v}^{\star} - \mathbf{v}(0)\right) \tag{29}$$

$$\mathbf{p}(t) = \mathbf{p}(0) - \delta(e^{-t/\delta} - 1)\mathbf{v}(0) + \left(t + \delta(e^{-t/\delta} - 1)\right)\mathbf{v}^{\star} \tag{30}$$

Hence, we can represent the AVO with a control obstacle by choosing the state $\mathbf{x} = [\mathbf{p}^{\mathrm{T}}\mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$, the control input $\mathbf{u} = \mathbf{v}^{\star}$, and

$$A = \begin{bmatrix} 0 & I \\ 0 & -\frac{1}{\delta}I \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ \frac{1}{\delta}I \end{bmatrix}, \qquad \mathbf{c} = \mathbf{0}$$

$$C = [I \quad 0], \qquad \mathbf{d} = \mathbf{0}$$

Solving equation (7), we find

$$F(t) = \begin{bmatrix} 1 & -\delta(e^{-t/\delta} - 1) \\ 0 & e^{-t/\delta} \end{bmatrix}, \quad G(t) = \begin{bmatrix} t + \delta(e^{-t/\delta} - 1) \\ 1 - e^{-t/\delta} \end{bmatrix},$$
$$\mathbf{h}(t) = \mathbf{0}$$

5.3. CCO

CCO (Ruffli et al., 2013) generalizes the previously mentioned AVO algorithm by defining the $(n + 1)$th derivative of position as the low-level control input where

$$\mathbf{p}^{(n+1)} = -c_n\mathbf{p}^{(n)} - \ldots - c_2\ddot{\mathbf{p}} + c_1(\mathbf{v}^{\star} - \dot{\mathbf{p}}) \tag{31}$$

where the superscript (n) represents the nth derivative of the term and $\mathbf{v}^{\star}$ is the target velocity that is a high-level control input.

The low-level control input is an input given directly to the robot which determines the next state through the equations of motion. The high-level control input abstracts the low-level control input to velocity through a controller. This abstraction to velocity as a control input is common in reciprocal collision avoidance and we discuss its use in our method in Section 6.1.

For example, controlling the jerk of a robot $\mathbf{p}^{(3)}$ is shown in full in Ruffli et al. (2013).

The state is $\mathbf{x} = [\mathbf{p}^{\mathrm{T}}\dot{\mathbf{p}}^{\mathrm{T}} \ldots (\mathbf{p}^{(n)})^{\mathrm{T}}]^{\mathrm{T}}$ and the control input is $\mathbf{u} = \mathbf{v}^{\star}$. Solving for the state equation gives

$$A = \begin{bmatrix} 0 & I & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & I \\ 0 & -c_1 I & \ldots & -c_n I \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ c_1 \end{bmatrix} \quad \mathbf{c} = \mathbf{0}$$

$$C = [I \quad 0 \quad \ldots \quad 0], \qquad \mathbf{d} = \mathbf{0}$$

5.4. LQR-obstacles

LQR-obstacles by Bareiss and van den Berg (2013) provide a method for reciprocal collision avoidance for homogeneous systems of robots with the same arbitrary linear equations of motion. The equations of motion of each robot are

$$\dot{\mathbf{x}} = \tilde{A}\mathbf{x} + \tilde{B}\mathbf{u} + \tilde{\mathbf{c}} \tag{32}$$

An LQR controller is used to obtain the low-level input from the high-level input $\mathbf{v}^{\star}$ using the control law

$$\mathbf{u} = -L\mathbf{x} + E\mathbf{v}^{\star} + \ell \tag{33}$$

By substituting equation (33) into equation (32), the closed-loop equations of motion are given as

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{v}^{\star} + \mathbf{c} \tag{34}$$

where

$$A = \tilde{A} - \tilde{B}L, B = \tilde{B}E, \mathbf{c} = \tilde{B}\ell + \tilde{\mathbf{c}} \tag{35}$$

Along with equation (34), the control obstacle is fully defined for a given C and $\mathbf{d}$ that extract the position from the state, similar to equation (5).

6. Non-homogeneous, non-linear equations of motion

The previous discussion defined a generalized method for reciprocal collision avoidance using control obstacles $\mathcal{UO}_{ij}$ for sets of robots with the same linear equations of motion. We present the extension of these methods for robots in

non-homogeneous systems, that is, different types, with possibly non-linear equations of motion.

Given the equations of motion of robots i and j (see equation (2)), we can approximate the relative position $\mathbf{p}_{ij}(t)$ given each robot's current state $\mathbf{x}_i$ and $\mathbf{x}_j$ as well as constant control inputs $\mathbf{u}_i$ and $\mathbf{u}_j$ through a first-order Taylor approximation about the current control input:

$$\mathbf{p}_{ij}(t) \approx \mathbf{q}_i[\mathbf{g}_i(t, \mathbf{x}_i, \mathbf{u}_i)] - \mathbf{q}_j[\mathbf{g}_j(t, \mathbf{x}_j, \mathbf{u}_j)] + \frac{\partial(\mathbf{q}_i \circ \mathbf{g}_i)}{\partial \mathbf{u}_i}(t, \mathbf{x}_i, \mathbf{u}_i)\Delta\mathbf{u}_i - \frac{\partial(\mathbf{q}_j \circ \mathbf{g}_j)}{\partial \mathbf{u}_j}(t, \mathbf{x}_j, \mathbf{u}_j)\Delta\mathbf{u}_j \tag{36}$$

In the spirit of equation (25), we make the assumption that $\Delta\mathbf{u}_i \approx \Delta\mathbf{u}_{ij}/2$ and $\Delta\mathbf{u}_j \approx -\Delta\mathbf{u}_{ij}/2$ such that each robot is required to take half the responsibility for avoiding pairwise collisions. For this assumption to be realistic, we require that the control input of both robots be of the same "type", for example a desired velocity, which we will discuss in Section 6.1. We can then re-write equation (36) as

$$\mathbf{p}_{ij}(t) \approx \hat{\mathbf{p}}_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j) + J_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j)\Delta\mathbf{u}_{ij} \tag{37}$$

where

$$\hat{\mathbf{p}}_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j) = \mathbf{q}_i[\mathbf{g}_i(t, \mathbf{x}_i, \mathbf{u}_i)] - \mathbf{q}_j[\mathbf{g}_j(t, \mathbf{x}_j, \mathbf{u}_j)] \tag{38}$$

$$J_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j) = \frac{1}{2}\left(\frac{\partial(\mathbf{q}_i \circ \mathbf{g}_i)}{\partial \mathbf{u}_i}(t, \mathbf{x}_i, \mathbf{u}_i) + \frac{\partial(\mathbf{q}_j \circ \mathbf{g}_j)}{\partial \mathbf{u}_j}(t, \mathbf{x}_j, \mathbf{u}_j)\right) \tag{39}$$

Given the definitions of equations (38) and (39), the control obstacle can be given similar to before as

$$\mathcal{UO}_{ij} = \bigcup_{0<t<\tau} J_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j)^{-1}(\mathcal{O}_{ij} \oplus \{-\hat{\mathbf{p}}_{ij}(t, \mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j)\}) \tag{40}$$

The methods for performing reciprocal collision avoidance with this new control obstacle formulation (equation (40)) are identical to those defined in Sections 4.3 and 4.4.

6.1. Higher-level control input

We have presented a method for reciprocal collision avoidance for robots in non-homogeneous systems with general, non-linear equations of motion. In doing so, we have made three key assumptions, which are:

(I) The control input remains constant over finite time τ;
(II) The robots observe each other's state and control input;
(III) The robots have the same type of input, equal in dimension to the workspace.

Many robots have control inputs which violate some or all of these assumptions. Let us consider a car-like robot with control inputs of acceleration at the rear axle and the steering angle. It is not reasonable to assume that these remain constant for long periods of time as required by (I). Of course, it will not remain constant because it changes every sensing-action cycle, but at least we want the constant assumption to give a reasonable estimate of the future motion of the other robots. For low-level control inputs that can change quickly (unlike a goal velocity), it cannot be assumed that a constant control input gives a reasonable estimate. It is also unreasonable to assume that these low-level control inputs can be observed by the other robots, violating (II). Performing reciprocal collision avoidance between a car-like robot and a differential-drive robot would violate (III).

For these reasons, we implement a controller which abstracts the low-level control inputs to a high-level control input, such as a target velocity, similar to Van den Berg et al. (2012), Bareiss and van den Berg (2013), and Ruffli et al. (2013). Abstracting to a high-level input makes the assumptions reasonable for most mobile robots. A target velocity typically remains approximately constant over long periods of time. A velocity is inherently equal to the dimension of the workspace. Lastly, it is reasonable to assume the current velocity of other robots can be observed, and it can be assumed the target velocity is approximately equal to the current velocity.

7. Results

We performed both simulations and physical experiments to verify the performance of the algorithm. In this section, we present the equations of motion for the robots used in the simulations and physical experiments as well as the result from those experiments. Each robot presented uses a controller to define a target velocity as a higher-level control input.

7.1. Robot dynamics

7.1.1. Differential-drive robot. We implemented a differential-drive robot in both simulation and experiments. We used the kinematic model with a three-dimensional state consisting of the two-dimensional position and the orientation (x, y, θ). The low-level control inputs are the left and right wheel velocities (v_r, v_l). The equations of motion are given as

$$\dot{x} = (v_r + v_l)\cos(\theta)/2 \tag{41}$$

$$\dot{y} = (v_r + v_l)\sin(\theta)/2 \tag{42}$$

$$\dot{\theta} = (v_r - v_l)/\ell \tag{43}$$

where ℓ is the distance between the wheels.

The low-level control inputs v_r and v_l are abstracted to a high-level input $\mathbf{v}^\star$ through a controller where

$$v_r = \|\mathbf{v}^\star\| + \ell k(\angle\mathbf{v}^\star - \theta)/2 \tag{44}$$

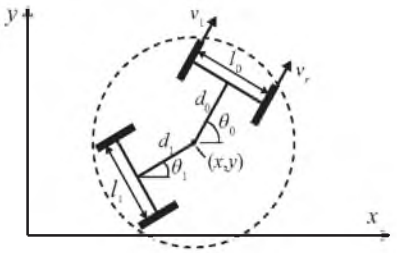

Fig. 4. The configuration of a differential-drive robot pulling a trailer with the origin (x, y) considered to be the point of connection between the robot and the trailer.

$$v_l = \|\mathbf{v}^\star\| - \ell k(\angle\mathbf{v}^\star - \theta)/2 \tag{45}$$

where $\angle\mathbf{v}^\star$ is the angle the target velocity makes with the positive x-axis and k is a controller gain. Substituting the new high-level control inputs from
'equations (44) and (45) into equations (41) to (43) gives

$$\dot{x} = \|\mathbf{v}^\star\| \cos(\theta), g t \dot{y} = \|\mathbf{v}^\star\| \sin(\theta), g t \dot{\theta} = k(\angle\mathbf{v}^\star - \theta) \tag{46}$$

7.1.2. Differential-drive with off-axle trailer We implemented a differential-drive robot pulling a trailer in both simulation and experiments. The equations of motion were adapted from Lee et al. (2004) which uses a car with an off-axle trailer. For the configuration given in Figure 4, the state is given as $(x, y, \theta_0, \theta_1)$ and the equations of motion are given as

$$\dot{x} = (v_r + v_l)\cos(\theta_0)/2 + \dot{\theta}_0 d_0 \sin(\theta_0) \tag{47}$$

$$\dot{y} = (v_r + v_l)\sin(\theta_0)/2 - \dot{\theta}_0 d_0 \cos(\theta_0) \tag{48}$$

$$\dot{\theta}_0 = (v_r - v_l)/\ell_0 \tag{49}$$

$$\dot{\theta}_1 = \left(\frac{v_r + v_l}{2}\sin(\theta_0 - \theta_1) - \dot{\theta}_0 d_0 \cos(\theta_0 - \theta_1)\right)/d_1 \tag{50}$$

where d_0, d_1, and ℓ_0 are the parameters shown in Figure 4.

We implemented a controller such that

$$\frac{v_r + v_l}{2} = \|\mathbf{v}^\star\|, \qquad \dot{\theta}_0 d_0 = k(\angle\mathbf{v}^\star - \theta_0) \tag{51}$$

which when substituting equation 51 into equations (47) to (50) gives

$$\dot{x} = \|\mathbf{v}^\star\| \cos(\theta_0) + k(\angle\mathbf{v}^\star - \theta_0)\sin(\theta_0) \tag{52}$$

$$\dot{y} = \|\mathbf{v}^\star\| \sin(\theta_1) - k(\angle\mathbf{v}^\star - \theta_0)\cos(\theta_0) \tag{53}$$

$$\dot{\theta}_0 = k(\angle\mathbf{v}^\star - \theta_0)/d_0 \tag{54}$$

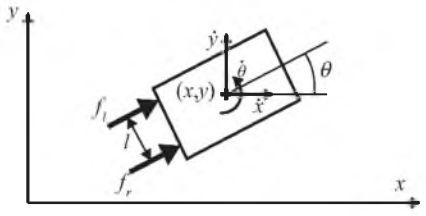

Fig. 5. The model of the hovercraft-like robot implemented in simulation. The two thrusters are shown as f_r and f_l with the distance between them as ℓ. The center position and orientation are shown.

$$\dot{\theta}_1 = (\|\mathbf{v}^\star\| \sin(\theta_0 - \theta_1) - k(\angle\mathbf{v}^\star - \theta_0)\cos(\theta_0 - \theta_1))/d_1 \tag{55}$$

7.1.3. Car-like robot. We implemented a car-like robot in simulation and physical experiments. We used a four-dimensional state consisting of the two-dimensional position, the orientation, and the speed (x, y, θ, v). The low-level control inputs are the acceleration at the rear axle and the steering curvature (a, κ) where equations of motion are defined at the midpoint by

$$\dot{x} = v\cos(\theta) - \ell v \kappa \sin(\theta)/2 \tag{56}$$

$$\dot{y} = v\sin(\theta) + \ell v \kappa \cos(\theta)/2 \tag{57}$$

$$\dot{\theta} = v\kappa \tag{58}$$

$$\dot{v} = a \tag{59}$$

where ℓ is the distance between the front and rear wheels. Deriving the equations of motion in terms of the midpoint of the robot, rather than at the midpoint along the rear axle keeps the enclosing disc as small as possible.

We implemented a proportional controller to determine the low-level control inputs in terms of the target velocity $\mathbf{v}^\star$:

$$a = k_0(\|\mathbf{v}^\star\| - v), \qquad \kappa = k_1 \ell(\angle\mathbf{v}^\star - \theta)/v \tag{60}$$

where k_0 and k_1 are proportional controller gains. Substituting equation 60 into equations (56) to (59) gives

$$\dot{x} = v\cos(\theta) - k_1(\angle\mathbf{v}^\star - \theta)\sin(\theta)/2 \tag{61}$$

$$\dot{y} = v\sin(\theta) + k_1(\angle\mathbf{v}^\star - \theta)\cos(\theta)/2 \tag{62}$$

$$\dot{\theta} = k_1(\angle\mathbf{v}^\star - \theta) \tag{63}$$

$$\dot{v} = k_0(\|\mathbf{v}^\star\| - v) \tag{64}$$

7.1.4. Hovercraft. We implemented a simulated hovercraft-style robot with two thrusters as seen in Figure 5. The hovercraft's state is given as the two-dimensional position, the heading, the two-dimensional velocity, and the rate of change of the heading $(x, y, \dot{x}, \dot{y}, \theta, \dot{\theta})$. Given the forces

provided by thrusters (f_r, f_l), the midpoint equations of motion are given as

$$\ddot{x} = ((f_r + f_l)\cos(\theta) - b_t\dot{x})/m \tag{65}$$

$$\ddot{y} = ((f_r + f_l)\sin(\theta) - b_t\dot{y})/m \tag{66}$$

$$\ddot{\theta} = \left((f_r - f_l)\ell/2 - b_r\dot{\theta}\right)/\iota \tag{67}$$

where m is the mass of the robot, ι is the robot's rotational inertia, b_t is the translational friction coefficient, and b_r is the rotational friction coefficient.

Including a proportional-derivative controller to solve for control inputs as a function of the target velocity $\mathbf{v}^\star$ gives

$$(f_r + f_l) = k_0 m(\|\mathbf{v}^\star\| - |\mathbf{v}|) \tag{68}$$

$$(f_r - f_l)\ell/2 - k_1\iota(\angle\mathbf{v}^\star - \theta) - k_2\dot{\theta} \tag{69}$$

where k_0 and k_1 are controller gains, $\mathbf{v} = (\dot{x}, \dot{y})$ is the velocity, and $\angle\mathbf{v}^\star$ is the angle the target velocity makes with the x-axis. Substituting equations (68) and (69) into equations (65) to (67) gives

$$\ddot{x} - k_0(\|\mathbf{v}^\star\| - \|\mathbf{v}\|)\cos(\theta) - b_t\dot{x}/m \tag{70}$$

$$\ddot{y} = k_0(\|\mathbf{v}^\star\| - \|\mathbf{v}\|)\sin(\theta) - b_t\dot{y}/m \tag{71}$$

$$\ddot{\theta} = k_1(\angle\mathbf{v}^\star - \theta) - (k_2 + b_r/\iota)\dot{\theta} \tag{72}$$

7.2. Simulation setup and implementation details

We performed simulations on a desktop machine running Windows 7 Professional 64-bit with an Intel i7-2600 CPU (3.40 GHz) and 8 GB RAM. The simulations were developed in a Visual Studio C++ environment. The frequency of the sensing-action control sequence was 10 Hz. The equations of motion were discretized using Runge–Kutta integration at 0.1 s time-steps.

The relative geometry $\mathcal{O}_{ij}$ was approximated using a set of 16 points uniformly sampled around a circle of the robots' combined radii. The control obstacle can then be approximated by performing the operations in equation (14) on the generated set of points for each time-step up to the time horizon τ. The convex hull of the control obstacle was computed using the Boost library (Dawes et al., 2009). Upon determining the halfplanes, the RVO2-2D library (Van den Berg et al., 2009) was used to compute the new control input through a convex optimization method.

The differential-drive robots had bounding circle radii of 0.3 m. The car-like robots had radii of 0.45 m. The hovercraft robots had radii of 0.47 m. The differential-drive robots with trailers had radii of 0.45 m. The desired speed of the robots during the simulations was 0.3 m/s.

Unless otherwise noted, we used a time horizon of $\tau = 7$ s during simulations. This value was determined experimentally. We found the selection of the time horizon to have a significant impact on the performance of our algorithm. Too short a time horizon can lead to a "late" reaction from the robots. This can lead to situations where a rather large change in control input is necessary to avoid collisions. If this large input violates the control input constraints on the robot, the collision-avoiding input is not obtainable and a collision can occur. On the other hand, selecting τ to be too large has a negative effect as well. The halfplanes from equation (18) become more restrictive as τ increases, possibly leading to no solution for equation (26). The time horizon is an empirical term that is situation-dependent, which is a limitation of our algorithm. We note that for the specific case of single integrator dynamics, Gal et al. (2009) have performed systematic analysis on the optimal value of the time horizon.

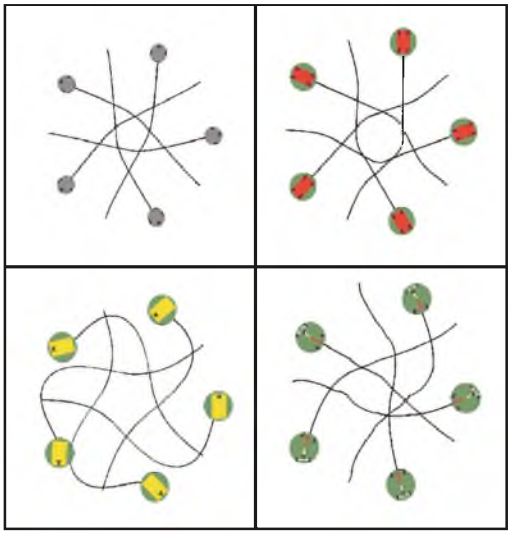

Fig. 6. Simulations where five robots avoid collisions while crossing the workspace. Top left: differential-drive robots. Top right: car-like robots. Bottom left: hovercraft-like robots. Bottom right: differential-drive robots with off-axis trailers.

7.3. Simulation results

We performed a variety of simulations to validate our approach. One set of simulations consisted of groups of five robots, where each robot type was simulated separately as shown in Figure 6. We ran simulations with all the robot types included. One such simulation included two of each type, eight in total. A selection of screenshots is shown in Figure 7. We included a simulation where two groups of four tried to cross the workspace as shown in Figure 8. We also performed a simulation where a group of four passive robots cross the workspace in a vertical line while a group of four active robots cross in the opposite direction. The four passive robots do not update their control input based on the positions of the other robots. Selected screenshots of this are shown in Figure 9. We performed a simulation

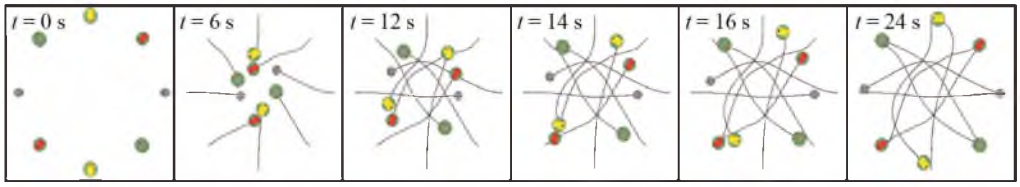

Fig. 7. A simulation that contains eight robots: two differential-drive (red discs), two differential-drives with trailers (red and white), two car-like robots (red rectangles), and two hovercrafts (yellow rectangles). They begin on a circle and cross the circle to finish on the side opposite their starting positions. Six screen shots from the simulation with the individual robot paths are shown.

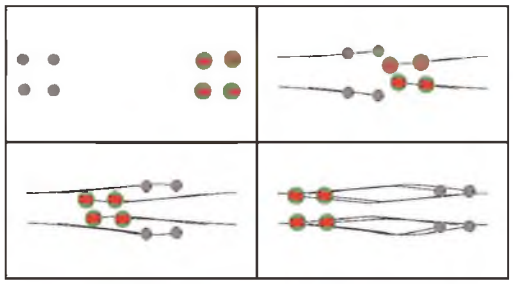

Fig. 8. Two groups of four robots crossing the workspace while avoiding collisions with each other.

with 100 robots moving from random starting positions to random goal positions. This simulation can be seen in the video found at http://arl.cs.utah.edu/research/grca/.

In physical experiments, it cannot be assumed that the robots can have perfect state estimation of the other robots. A series of simulations were performed to demonstrate our algorithm's robustness in the presence of noise. A group of eight differential-drive robots were initialized in a circle and their goal was to cross to the antipodal position. In these simulations, we introduced artificial noise by adding a normal random variable to the position components of $\mathbf{x}_j$ in equation (14). Increasing the robot's radius is a common practice in reciprocal collision avoidance methods, and therefore we increased our robots' radii by 10% (0.03 m). Using a time horizon of $\tau = 5$ s, no collisions were observed until the standard deviation of the sensed position was 0.07 m. We repeated the experiments with the radii increased by 25% (0.075 m) and observed collisions at a standard deviation of 0.15 m. Similar simulations were run for the same scenario with two of each robot type (differential-drive, differential-drive with a trailer, car-like, and hovercraft). In this case, the algorithm was less robust to noise with collisions resulting from standard deviations of 0.02 m and 0.04 m for bounding radii increases of 10% and 25%, respectively. This is likely due to the more constrained input space $\mathcal{U}_{ij}$ for the more complicated dynamics as well as our use of very simple controllers for complex equations of motion. These experiments, as we expected, suggest that for larger noise values a larger increase in the bounding circle can be used. However, too large a bounding circle makes for extremely conservative actions which may be too limiting for a given robot's control input constraints.

In order to quantify the speed of our algorithm, we calculated the per-time-step-per-robot average computation time, that is, the time it takes for one robot to determine its set of safe changes in velocity with respect to every other robot in a single sensing-action cycle. As can be seen in Figure 10, this quantity is linear with respect to the number of robots, as expected. In simulation, we found that it is possible for over 100 non-homogeneous, non-linear robots to perform reciprocal collision avoidance at real-time computational rates for a time horizon of $\tau = 20$ s with a simulation frequency of 10 Hz. At higher frequencies, for example, 20 Hz, the trends in Figure 10 would have a slope of twice that for 10 Hz, due to the doubled frequency during the integration in equation (15). As can be seen, the computation time also shows a linear trend with the value

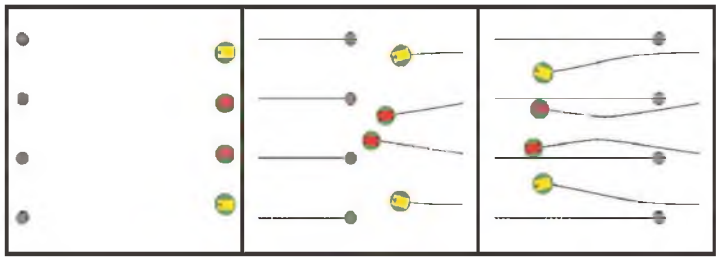

Fig. 9. A case where two *active* car-like robots and two *active* hovercraft robots cross the workspace from right to left as four *passive* differential-drive robots cross from left to right. The paths the robots take are drawn and it can be seen that the car-like and hovercraft robots make the necessary adjustments to avoid collisions with each other and the differential-drive robots.

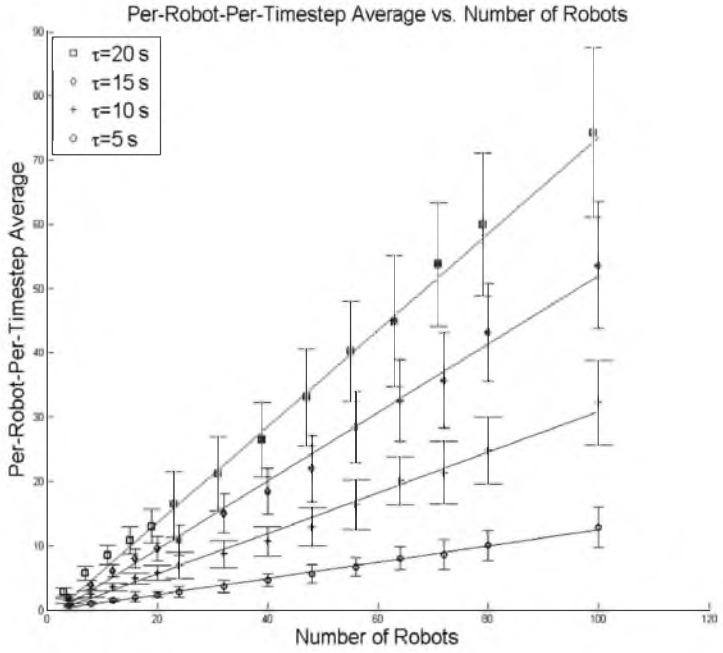

Fig. 10. Timing calculations were made for a number of experiments shown above. The data and the first-order fit are shown. For a sensing-action cycle frequency of 10 Hz, a single time-step is 0.1 s and our algorithm can produce real-time results for over 100 robots.

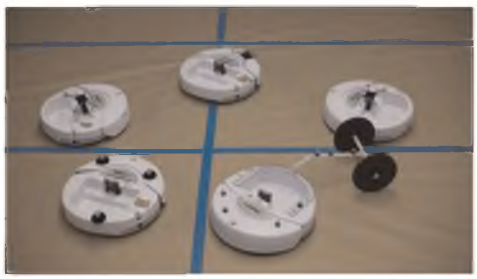

Fig. 11. An image taken while performing the physical experiments of our reciprocal collision avoidance algorithm.

of the time horizon, as twice as large a time horizon results in twice as complex a control obstacle (at least in the way we implemented its construction).

7.4. Experiments

The physical experiments were performed using the Robot Operating System (ROS) software platform. A motion-capture environment was used to estimate the positions and orientations of the robots. Similar to the simulations, the control obstacles were approximated using sets of 16 points uniformly sampled around the robots' bounding circles. While the state estimates of the robots are performed by the motion capture system, the algorithm is developed as a decentralized method where on-board state estimation could be implemented in the future.

For the experiments, we used six iRobot Creates. Three of the Creates were used as differential-drive robots, two were used to simulate car-like motion by restricting their minimum turning radius, and the sixth Create had a custom trailer mounted to it. The Creates have a radius of 0.335 m. The trailer axle is located 0.25 m behind the center of the Create. The robots were driven with a desired speed of 0.2 m/s. Their maximum speed possible is 0.5 m/s. The experiments were performed with a frequency of 50 Hz and a time horizon of 3.5 s.

Due to the stochastic nature of the experiments from modeling and sensor error, the robots' bounding circle was increased by 25%. Less accurate models or less accurate sensors could require a further increase in the radius. At times the robots can be seen moving back and forth between each other in a form of "reciprocal dance" due to sensing noise. This phenomenon has been more thoroughly studied by Conroy et al. (2014).

During the experiments, we recorded the desired velocity before the control obstacles algorithm was performed as well as the collision-free target velocity resulting from the control obstacles. To further quantify the experimental results, we determined the Euclidean norm between the desired and the calculated target velocities, representing the change in input from the algorithm. From approximately 5500 data points the mean and standard deviation of the set were found to be 0.0794 m/s and 0.128 m/s, respectively.

We ran experiments similar to the simulations with the robots crossing through the center of the workspace and avoiding collisions as shown in Figure 11. We also

performed experiments of situations more applicable to real-world scenarios where the robots were divided into two groups crossing the workspace. Videos of the experiments and simulations can be found at http://arl.cs.utah.edu/research/grca/.

8. Conclusions

Previously, reciprocal collision avoidance has been applied to a variety of robotic systems with both linear and special cases of non-linear equations of motion. In these approaches, the fact that all the robots were the same type allowed for a relative state formulation to be used. However, for non-homogeneous systems, that is, systems of robots that are different types, this is not possible.

In this paper, we presented a unified method for reciprocal collision avoidance of non-homogeneous systems of robots with non-linear equations of motion. In order to do so, we presented the control obstacle for homogeneous systems of robots with linear equations of motion. We then showed how the control obstacle generalizes previous reciprocal collision avoidance methods and provided examples of how previous methods fit into our framework. More specifically, we showed VO (Fiorini and Shiller, 1998), AVO (Van den Berg et al., 2012), CCO (Ruffli et al., 2013), and LQR-obstacles (Bareiss and van den Berg, 2013). Finally, we extended control obstacles for use with non-linear equations of motion and/or non-homogeneous systems. In our simulations and physical experiments, we saw that our algorithm was able to provide smooth, collision-free motion for all robots in the environment.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

Alonso-Mora J, Breitenmoser A, Beardsley P, et al. (2012) Reciprocal collision avoidance for multiple car-like robots. In: *IEEE international conference on robotics and automation*.

Alonso-Mora J, Breitenmoser A, Rufli M, et al. (2010) Optimal reciprocal collision avoidance for multiple non-holonomic robots. In: *Proceedings of the 10th international symposium on distributed autonomous robotic systems*.

Bareiss D and van den Berg J (2013) Reciprocal collision avoidance for robots with linear dynamics using LQR-obstacles. In: *IEEE international conference on robotics and automation*.

Conroy P, Bareiss D and Beall M (2014) 3-D reciprocal collision avoidance on physical quadrotor helicopters with on-board sensing for relative positioning. Available at: http://arxiv.org/abs/1411.3794.

Dawes B, Abrahams D and Rivera R (2009) Boost C++ libraries. Available at: http://www.boost.org.

Fiorini P and Shiller Z (1998) Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research* 17: 760–772.

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

Gal O, Shiller Z and Rimon E (2009) Efficient and safe on-line motion planning in dynamic environments. In: *IEEE international conference on robotics and automation*.

Giese A, Latypov D and Amato N (2014) Reciprocally-rotating velocity obstacles. In: *IEEE international conference on robotics and automation*.

Hsu D, Kindel R, Latombe J, et al. (2002) Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research* 21(3): 233–255.

Kluge B and Prassler E (2004) Reflective navigation: Individual behaviors and group behaviors. In: *IEEE international conference on robotics and automation*.

Lalish E and Morgansen K (2012) Distributed reactive collision avoidance. *Autonomous Robots* 32(3): 207–226.

Lee JH, Chung W, Kim M, et al. (2004) A passive multiple trailer system with off-axle hitching. *International Journal of Control, Automation, and Systems* 2(3): 289–297.

Petti S and Fraichard T (2005) Safe motion planning in dynamic environments. In: *IEEE RSJ international conference on intelligent robots and systems*.

Ruffli M, Alonso-Mora J and Siegwart R (2013) Reciprocal collision avoidance with motion continuity constraints. *IEEE Transactions on Robotics* 29(4): 899–911.

Shiller Z, Large F and Sekhavat S (2001) Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In: *IEEE international conference on robotics and automation*.

Shiller Z, Prasanna R and Salinger J (2008) *A unified approach to forward and lane-change collision warning for driver assistance and situational awareness*. Technical Report 2008-01-0204, Society of Automotive Engineers, Warrendale, PA.

Snape J, van den Berg J, Guy S, et al. (2010) Smooth and collision-free navigation for multiple robots under differential-drive constraints. In: *IEEE international conference on intelligent robots and systems*.

Van den Berg J, Guy S, Lin M, et al. (2009) Reciprocal *n*-body collision avoidance. In: *Proceedings of the international symposium of robotics research*.

Van den Berg J, Ling M and Manocha D (2008) Reciprocal velocity obstacles for real-time multi-agent navigation. In: *IEEE international conference on robotics and automation*.

Van den Berg J, Snape J, Guy S, et al. (2012) Reciprocal collision avoidance with acceleration-velocity obstacles. In: *IEEE international conference on robotics and automation*.

Wilkie D, van den Berg J and Manocha D (2009) Generalized velocity obstacles. In: *IEEE international conference on robotics and systems*.

CHAPTER 3

STOCHASTIC AUTOMATIC COLLISION AVOIDANCE FOR TELE-OPERATED UNMANNED AERIAL VEHICLES

The work in this chapter addresses Objective 2, Task 1, where a stochastic (feedforward model-based) automatic collision avoidance algorithm was developed and applied to tele-operated unmanned aerial vehicles. The theoretical and simulation results were published in the *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* in 2015 and the full paper is reprinted here with permission. More specifically, the paper develops the method to avoid collisions in the presence of uncertainty in the robot's motion model as well as the sensing of the obstacles' position. The algorithm is studied in simulation with varying levels of uncertainty, quantifying the true probability of collisions using this algorithm as well as comparing the performance of the algorithm to a deterministic system.

D. Bareiss, J. van den Berg, and K. K. Leang. "Stochastic Automatic Collision Avoidance for Tele-Operated Unmanned Aerial Vehicles," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Hamburg, Germany, Oct. 2015, pp. 4818-4825.

Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Stochastic Automatic Collision Avoidance for Tele-Operated Unmanned Aerial Vehicles

Daman Bareiss Jur van den Berg Kam K. Leang

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

I. Introduction

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

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

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

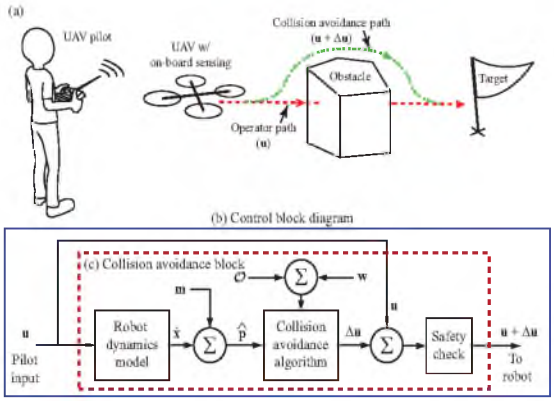

Fig. 1. Collision avoidance for tele-operated UAVs: (a) concept where UAV pilot controls the aircraft and provides a user input $\mathbf{u}$. When a collision is detected, with on-board sensing and state estimation, the algorithm produces an input $\Delta\mathbf{u}$ that augments the pilot's input $\mathbf{u}$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

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

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

978-1-4799-9994-1/15/$31.00 ©2015 IEEE

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

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

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

II. Related Work

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

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

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

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

III. Problem Formulation

A. Notation

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

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

B. System equations, uncertainty, workspace, and obstacles

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

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

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

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

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

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

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

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

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

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

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

where

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

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

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

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

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

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

C. Problem Definition

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

$$\text{minimize: } \Delta\mathbf{u}^T R \Delta\mathbf{u} \tag{9}$$
$$\text{subject to: } p\left(\forall t \in [0, \tau] :: \mathcal{R}(\mathbf{x}(t)) \cap \mathcal{O} = \emptyset \mid P(t), Z\right) \leq \bar{p},$$

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

IV. A Stochastic Approach to Collision Avoidance

A. Approach

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

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

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

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

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

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

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

where

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

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

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

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

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

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

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

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

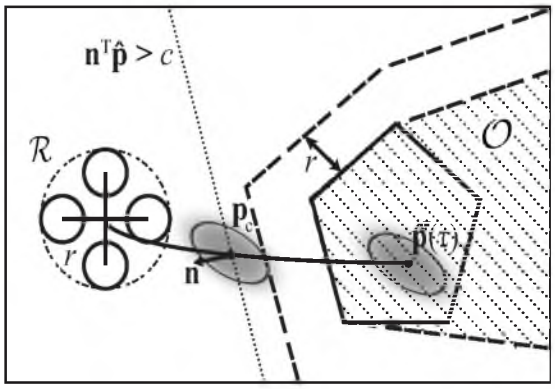

Fig. 2. Shown is a scenario where a quadrotor helicopter has uncertainty in both its trajectory estimation and the obstacle location. The a priori variance $P_c(\tau) + Z$ for a given obstacle normal $\mathbf{n}$ is represented as $\mathbf{n}^T(P_c(\tau) + Z)\mathbf{n}$. The collision point $\mathbf{p}_c$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

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

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

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

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

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

where

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

B. Handling Convex Edges and Corners Through Iteration

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

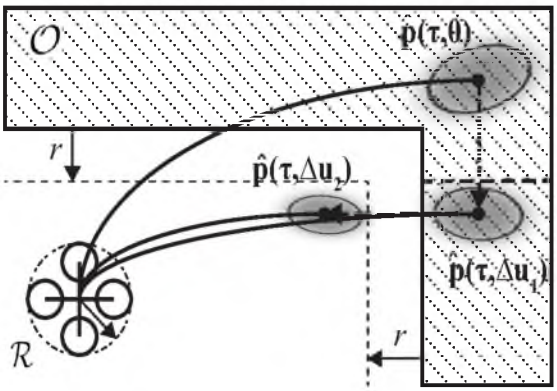

Fig. 3. The iterative process of the algorithm is shown. A trajectory is estimated from the original operator's input such that $\Delta\mathbf{u} = 0$ and is shown as $\hat{\mathbf{p}}(\tau, 0)$. The variance on this estimated position is shown as the gray ellipsoid located at $\hat{\mathbf{p}}(\tau,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

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

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

where

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

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

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

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

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

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

V. Simulation: Results and Discussion

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

A. Implementation Details

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

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

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

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

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

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

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

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

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

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

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

B. Results and Discussion

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

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

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

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

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

TABLE I
ERROR CALCULATIONS AS FRACTION OF RADIUS

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

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

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

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

while the third experiment used a larger covariance of

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

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

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

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

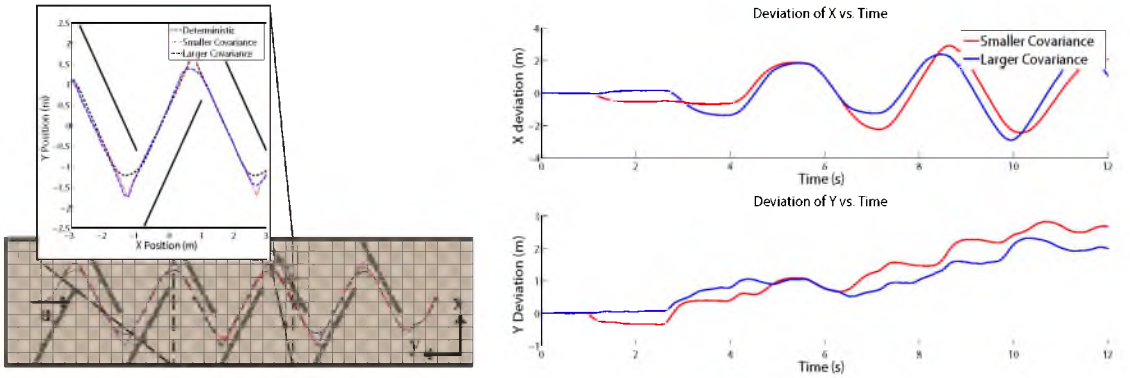

Fig. 4. An experiment was performed to compare the path of the quadcopter through the environment for deterministic collision avoidance and stochastic collision avoidance with two sets of variances [Eqs. (35)–(38)]. On the left, the resulting paths are given and a zoomed portion is given for clarity.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

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

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

VI. Conclusions and Future Work

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

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

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

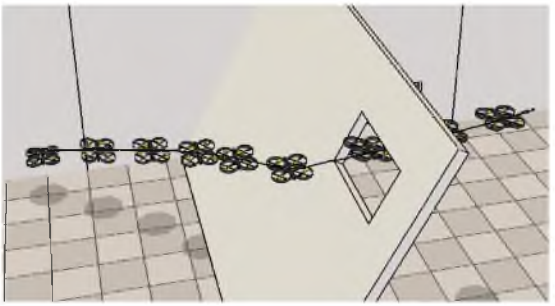

Fig. 5. A 3-dimensional example is shown where the quadrotor is steered towards a goal point through a window on a slanted wall. The window has tight clearance with respect to the robot. The height of the window is only 25% larger than the robot's diameter, however, the diameter is a conservative estimate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

VII. Acknowledgments

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

CHAPTER 4

ONBOARD MODEL-BASED AUTOMATIC COLLISION AVOIDANCE: APPLICATION IN REMOTELY PILOTED UNMANNED AERIAL VEHICLES

4.1 Introduction

Recent advances in unmanned aerial vehicles (UAVs) include improved flight times [1], [2], advanced flight control systems [3], and reduced development costs [4], and 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 [5], [6], search and rescue [7], precision farming [8], space exploration [9], traffic management [10], environmental monitoring [11], [12], telecommunication [13], and even entertainment [14]. 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 [15]–[19].

Despite recent advances in the design and development of UAVs, particularly hover-capable rotorcraft UAVs, the task of carefully navigating the UAV through a cluttered environment and avoiding a collision with nearby obstacles and humans remains a challenge [20], [21]. 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. The need for automatic collision avoidance technology for tele-operated UAVs is critical and necessary to allow pilots to focus on higher-priority tasks such as locating survivors.

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 multirotor 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 in [22], [23], 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.

Figure 4.1 shows the block diagram of the collision avoidance approach along with the newly proposed on-board sensing scheme. 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 are 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 lead to 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 is used) is 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 [24], the dynamic window approach [25], velocity obstacles [26], and vector field histograms (VFH) [27], the proposed approach is based upon local sensor information but can exploit global information as well. The approach also considers the uncertainty in the estimation and measurement process, and applies to the full (possibly nonlinear) robot dynamics.

The remainder of this chapter is structured as follows. A detailed summary of related work and the comparison of this work to similar techniques is presented in Sec. 4.2. A detailed summary of the stochastic collision avoidance algorithm is given in Sec. 4.3, fol-

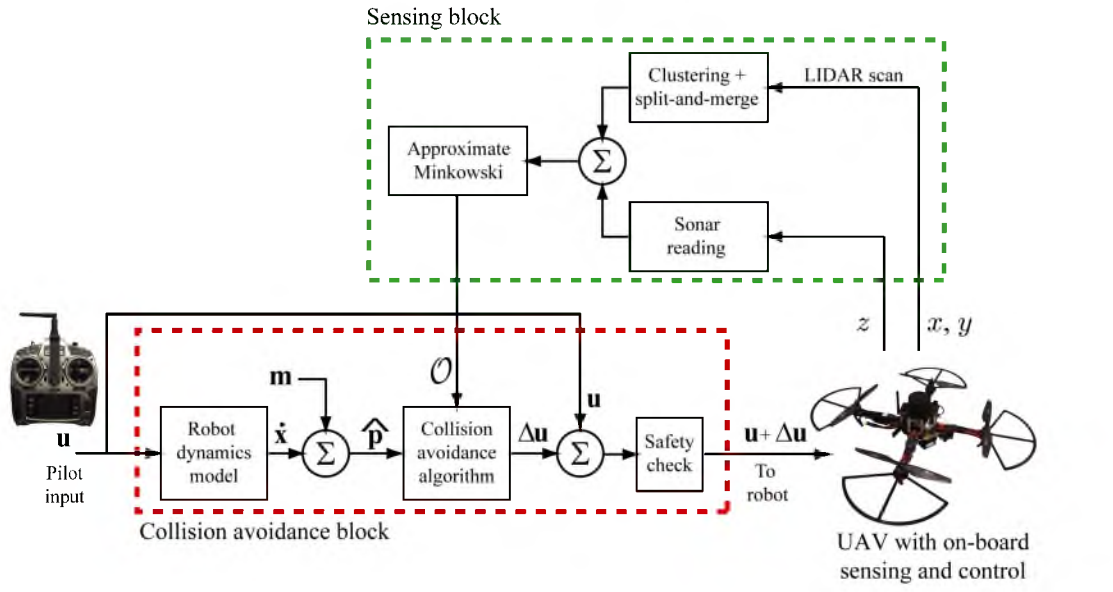

Figure 4.1: Collision avoidance for tele-operated UAVs: A UAV pilot controls the aerial vehicle and provides input $\mathbf{u}$. When a collision is detected with on-board sensing and state estimation, the algorithm produces an input $\Delta\mathbf{u}$ that augments the pilot's input $\mathbf{u}$ to 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.

lowed by a description of the custom-designed experimental UAV system with on-board computation and obstacle detection in Sec. 4.4. The experimental results and discussion are presented in Sec. 4.5. Finally, concluding remarks and a discussion of future work are presented in Sec. 4.6, and acknowledgments are found in Sec. 4.7.

4.2 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 [28], medical devices [29], and mobile service robots [27]. Some early methods include potential fields [24], the dynamic window approach [25], velocity obstacles [26], and vector field histograms (VFH) [27]. In general, collision avoidance methods can be classified into one of two main categories: global (motion planning methods) and local (reactive methods).

First, 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 [30]–[32]. In [33], 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 a 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 [34]. Other techniques deal with state uncertainty by exploiting dynamic programming [35].

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 [36]–[39]. The algorithm in this chapter 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 [26]. Typically, these algorithms avoid collisions by computing a given change in input for a current sensing-action cycle [34], [40], [41]. The approach in this chapter is intended for use with local sensor information, but it can also be applied in situations where global knowledge is provided. The formulation is applicable to the full robot dynamics and the method considers explicitly the uncertainty in the estimation and measurement process.

In [36], the FastSLAM algorithm [42] 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 [43], [44]. Vision has been applied to obstacle detection and avoidance in UAVs as well [45]–[48]. 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 [49]–[51].

4.3 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 [23]. First, the problem is formally defined below in Sec. 4.3.1, followed by details of the algorithm in Sec. 5.3.3. It is pointed out that the work in [22], [23] 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 Sec. 4.3.2.4 presents the details.

4.3.1 Problem Formulation

4.3.1.1 Notation

Throughout this chapter, vectors are denoted by boldface lower-case letters, for example $\mathbf{a}$. 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:

$$x\mathcal{A} = \{x\mathbf{a} \mid \mathbf{a} \in \mathcal{A}\}, \tag{4.1}$$

$$X\mathcal{A} = \{X\mathbf{a} \mid \mathbf{a} \in \mathcal{A}\}, \tag{4.2}$$

$$\mathcal{X} \oplus \mathcal{A} = \{\mathbf{x} + \mathbf{a} \mid \mathbf{x} \in \mathcal{X}, \mathbf{a} \in \mathcal{A}\}. \tag{4.3}$$

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

$$\mathbf{a} \sim \mathcal{N}(\bar{\mathbf{a}}, \Sigma). \tag{4.4}$$

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 $\mathbf{f} \in \mathcal{X} \times \mathcal{U} \to \mathbb{R}^m$,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{m}, \quad \mathbf{m} \sim \mathcal{N}(\mathbf{0}, M), \tag{4.5}$$

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 semidefinite.

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

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(0), \mathbf{u}, t). \tag{4.6}$$

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

$$\mathbf{x}(t) \sim \mathcal{N}(\hat{\mathbf{x}}(t), P(t)), \tag{4.7}$$

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

$$P(t) = \mathrm{E}[(\mathbf{x}(t) - \hat{\mathbf{x}}(t))(\mathbf{x}(t) - \hat{\mathbf{x}}(t))^T]. \tag{4.8}$$

The uncertainty at time t, Eq. (4.8), is found by solving the following differential equation:

$$\dot{P}(t) = A(t)P(t) + P(t)A(t)^T + M, \tag{4.9}$$

$$A(t) = \frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}}(\hat{\mathbf{x}}(t), \mathbf{u}(t)). \tag{4.10}$$

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

$$\mathcal{O} = \bigcup_{i=1}^{n} \mathcal{O}_i \oplus \{\mathbf{w}\}, \quad \mathbf{w} \sim \mathcal{N}(0, Z(\mathbf{x})), \tag{4.11}$$

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

$$\mathcal{R}(\mathbf{x}(t)) \cap \mathcal{O} \neq \emptyset. \tag{4.12}$$

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

$$\begin{aligned} &\text{minimize: } \Delta\mathbf{u}^T R \Delta\mathbf{u} \\ &\text{subject to:} \\ &\quad p\left(\forall t \in [0,\tau] :: \mathcal{R}(\mathbf{x}(t)) \cap \mathcal{O} = \emptyset \mid P(t), Z(\mathbf{x})\right) \leq \bar{p}, \end{aligned} \tag{4.13}$$

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.

4.3.2 Technical Approach

In the following, a model-based feedforward approach is presented for collision avoidance (see block diagram in Fig. 4.1).

4.3.2.1 Assumptions

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

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

$$\mathbf{p}(t) = C\mathbf{x}(t), \tag{4.14}$$

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

- 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 $\mathbf{p}$.

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

$$\hat{\mathbf{p}}(t, \Delta\mathbf{u}) \approx \hat{\mathbf{p}}^\star(t) + J(t)\Delta\mathbf{u}, \tag{4.15}$$

  where

$$\hat{\mathbf{p}}^\star(t) = C\mathbf{g}(\hat{\mathbf{x}}, \mathbf{u}, t), \; J(t) = C\frac{\partial \mathbf{g}}{\partial \mathbf{u}}(\hat{\mathbf{x}}, \mathbf{u}, t). \tag{4.16}$$

- 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.

4.3.2.2 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. (5.8) (see block diagram in Fig. 4.1). The variance on the predicted state is given in Eq. (4.8).

From Assumption 4.3.2.1, the mapping from the robot's state to its position also defines the variance on the robot's position, *i.e.*,

$$P_c(t) = CP(t)C^T. \tag{4.17}$$

From Assumption 4.3.2.1, 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 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 nonzero if

$$\mathcal{P}_{\text{collision}} = (\mathcal{N}(\hat{\mathbf{p}}^\star(\tau), P_c(\tau) + Z(\mathbf{x})) \cap \mathcal{O}) \neq \emptyset. \tag{4.18}$$

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.*,

$$p((\mathcal{R}(\mathbf{p}(\tau)) \cap \mathcal{O} \neq \emptyset) \mid (\mathcal{P}_{\text{collision}} \neq \emptyset)) \leq \bar{p}. \tag{4.19}$$

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

$$\mathbf{p}_c = \hat{\mathbf{p}}^\star(t_c), \tag{4.20}$$

where

$$t_c = \underset{t \in [0,\tau]}{\operatorname{argmin}} \{p((\mathcal{R}(\mathbf{p}(t)) \cap \mathcal{O} \neq \emptyset) \mid (\mathcal{P}_{\text{collision}} \neq \emptyset)) > \bar{p}\}. \tag{4.21}$$

The probabilities in Eqs. (4.19), (4.21) can be difficult to compute exactly. Therefore, the approximate solution is considered. Given a unit normal vector $\mathbf{n}$ of the obstacle $\mathcal{O}$

that points into the free workspace (see Fig. 4.2), 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. (4.21).

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 $\mathbf{p}_c$,

$$\mathcal{P}_{\text{collision}} \approx \mathcal{N}(\hat{\mathbf{p}}^\star(\tau), \mathbf{n}^T(P_c(\tau) + Z(\mathbf{x}))\mathbf{n}). \tag{4.22}$$

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 (4.19), 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

$$\forall t \in [0, \tau] :: \mathcal{R}(\hat{\mathbf{p}}^\star(t)) \cap (\mathcal{O} \oplus \{\hat{\sigma}\mathbf{n}\}) = \emptyset, \tag{4.23}$$

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

$$\hat{\sigma} = a\mathbf{n}^T\sqrt{P_c(\tau) + Z(\mathbf{x})}\mathbf{n}, \tag{4.24}$$

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

Equations (5.9) and (4.21) can now be approximated by the following simplified expression:

$$\mathbf{p}_c = \hat{\mathbf{p}}^\star\left(\underset{t\in[0,\tau]}{\text{argmin}}\{\mathcal{R}(\hat{\mathbf{p}}^\star(t)) \cap (\mathcal{O} \oplus \{\hat{\sigma}\mathbf{n}\}) \neq \emptyset\}\right), \tag{4.25}$$

where if the system has no uncertainty $\hat{\sigma} = 0$, then Eqs. (4.23),(4.25) are equivalent to the deterministic solution in [22].

Next, given Eqs. (4.23), (5.9), a linear constraint is defined on the position $\hat{\mathbf{p}}(\tau, \Delta\mathbf{u})$ of the robot at time τ (see Eq. (4.2))

$$\mathbf{n}^T\hat{\mathbf{p}}(\tau, \Delta\mathbf{u}) > \mathbf{n}^T\mathbf{p}_c. \tag{4.26}$$

Substituting Eq. (5.8) from Assumption 4.3.2.1, the constraint on the robot's position in Eq. (5.10) can be transformed into a constraint on its change in input $\Delta\mathbf{u}$

$$\mathbf{n}^T J(\tau)\Delta\mathbf{u} > \mathbf{n}^T(\mathbf{p}_c - \hat{\mathbf{p}}^\star(\tau)). \tag{4.27}$$

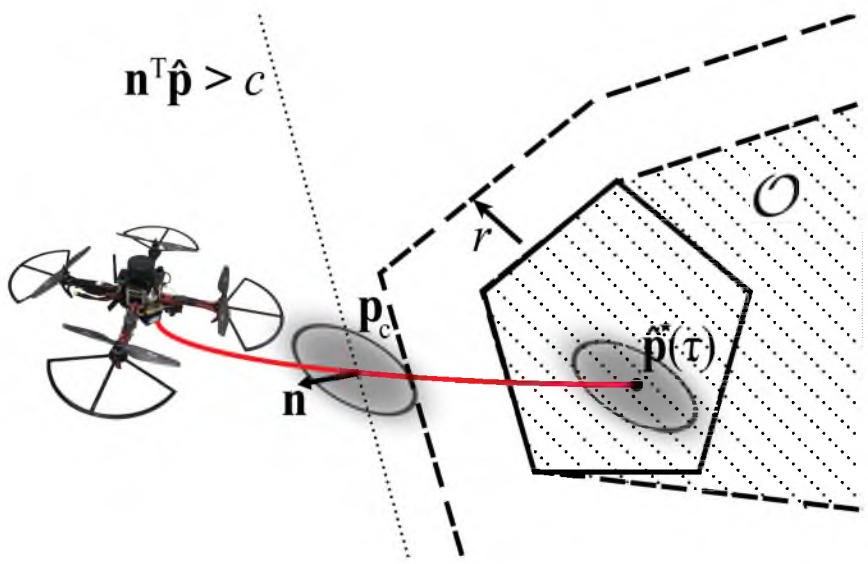

Figure 4.2: Shown is a scenario where a quadcopter UAV has uncertainty in both its trajectory estimation and the obstacle location. The a priori variance $P_c(\tau) + Z(\mathbf{x})$ for a given obstacle normal $\mathbf{n}$ is represented as $\mathbf{n}^T(P_c(\tau) + Z(\mathbf{x}))\mathbf{n}$. The collision 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 $c = \mathbf{n}^T\mathbf{p}_c$.

Equation (5.7) is approximated using Eq. (5.11) as

$$\begin{aligned} &\text{minimize: } \Delta\mathbf{u}^T R\Delta\mathbf{u} \\ &\text{subject to: } \mathbf{n}^T J(\tau)\Delta\mathbf{u} > \mathbf{n}^T(\mathbf{p}_c - \hat{\mathbf{p}}^{\star}(\tau)), \end{aligned} \tag{4.28}$$

where solving this convex optimization, such as is done by the RVO library in [52], provides a collision-free change in input $\Delta\mathbf{u}$ with the control input given to the robot as $\mathbf{u} + \Delta\mathbf{u}$.

4.3.2.3 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. 4.3. 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. (5.7) 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

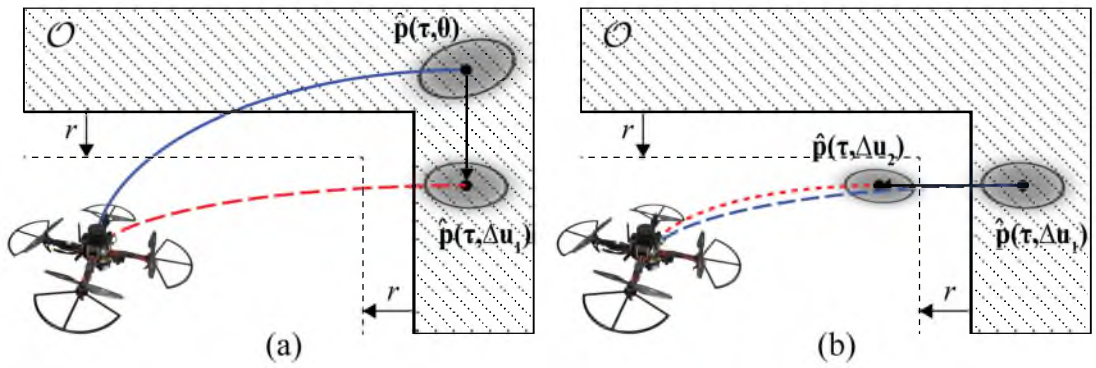

Figure 4.3: The iterative process of the collision avoidance algorithm: (a) A trajectory, $\hat{\mathbf{p}}(\tau, \mathbf{0})$, is estimated from the original operator's input such that $\Delta\mathbf{u} = \mathbf{0}$. The variance on this estimated position is shown as the gray ellipsoid located at $\hat{\mathbf{p}}(\tau, \mathbf{0})$. 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, also resulting in a collision, and the algorithm computes a new change in input $\Delta\mathbf{u}_2$ with respect to halfplane constraints defined by both $\hat{\mathbf{p}}(\tau, \mathbf{0})$ 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. 4.1).

$$\mathbf{a}_i^T \Delta\mathbf{u} > b_i, \tag{4.29}$$

where

$$\mathbf{a}_i^T = \mathbf{n}_i^T J_i(\tau), \tag{4.30}$$

$$b_i = \mathbf{n}_i^T (\mathbf{p}_{c,i} - \hat{\mathbf{p}}_i(\tau)). \tag{4.31}$$

The convex optimization problem in Eq. (5.12) can now be solved for all iterations i by the following:

$$\begin{aligned} &\text{minimize:} \Delta\mathbf{u}^T R \Delta\mathbf{u} \\ &\text{subject to:} \bigcap_{j=1}^{i} \{\mathbf{a}_j^T \Delta\mathbf{u} > b_j\}. \end{aligned} \tag{4.32}$$

Thus, every i^{th} iteration of the algorithm introduces an additional constraint to the convex optimization problem. After at most d iterations, the control input $\mathbf{u} + \Delta\mathbf{u}$ 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. 4.3. This iterative approach is performed during every sensing-action cycle of the robot.

This iterative approach aligns with the LP-type algorithm in [53]. 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 chapter 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.

4.3.2.4 Incorporating Yaw Control

For a hover-capable multirotor UAV, yaw is a redundant degree-of-freedom that can be held constant. This condition is further emphasized in Sec. 4.4.1.1. 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 Sec. 4.3.2.3, 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. (4.16).

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 4.3.2.1 or can violate Assumption 4.3.2.1, possibly leading to collisions.

4.4 The Experimental UAV System with Onboard Computation and Sensing

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

4.4.1 The Experimental Quadcopter UAV System

The collision avoidance and obstacle detection algorithms are implemented on a custom-designed physical quadrotor helicopter shown in Fig. 4.4. The UAV has a footprint of 75 cm from rotor-tip to rotor-tip. In Fig. 4.4, 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 LIDAR-Lite laser rangefinder from

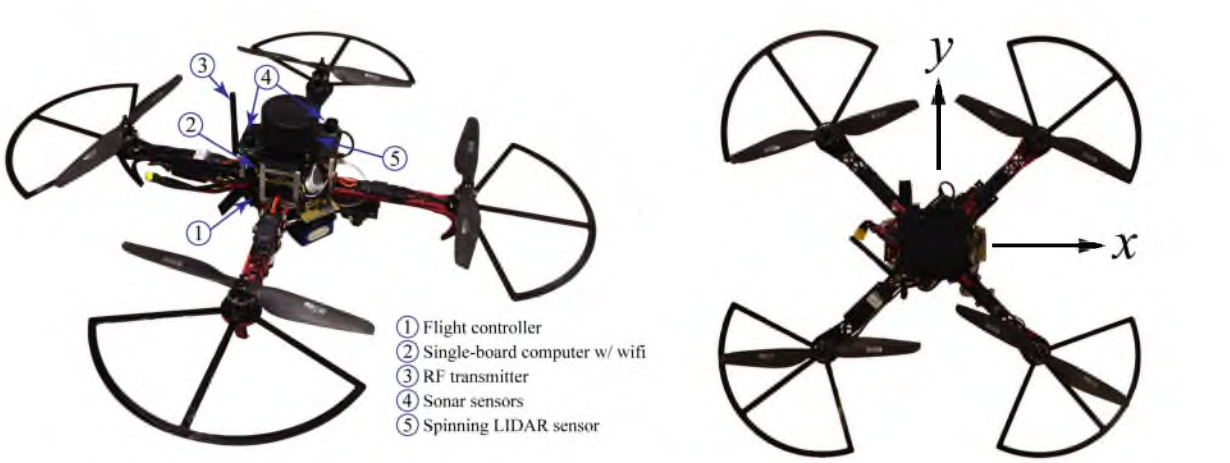

Figure 4.4: The custom-designed experimental quadcopter UAV system with onboard obstacle detection hardware shown on the left. The Odroid XU4, running the Robot Operating System (ROS), reads the sensor data and runs the algorithms onboard and provides commands to the Pixhawk autopilot to control the motion of the UAV. 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.

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 onboard single-board computer (Odroid XU4). 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 UAV.

4.4.1.1 Robot Dynamics

The Pixhawk autopilot contains onboard proportional-integral-derivative (PID) controllers to stabilize the attitude of the robot. The Pixhawk also uses a raw throttle command. This throttle command is calculated by a controller about the vertical velocity. The closed-loop model is used to calculate the feedforward trajectory estimate provided an estimate of the current state through an onboard 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 $\mathbf{w} \in \mathbb{R}^3$:

$$\mathbf{x} = [\mathbf{p}^T, \mathbf{v}^T, \mathbf{r}^T, \mathbf{w}]^T \in \mathcal{X}. \tag{4.33}$$

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 $w_z^\star$:

$$\mathbf{u} = [r_x^\star, r_y^\star, v_z^\star, w_z^\star]^T \in \mathcal{U}. \tag{4.34}$$

The equations of motion are given as

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{4.35}$$

$$\dot{\mathbf{v}} = R\left[0, 0, k_{\text{pv}}(v_z^\star - v_z)\right]^T - \mathbf{g}, \tag{4.36}$$

$$\dot{\mathbf{r}} = \mathbf{w}, \tag{4.37}$$

$$\dot{\mathbf{w}} = \begin{bmatrix} k_{\text{px}}(r_x^\star - r_x) - k_{\text{dx}} w_x \\ k_{\text{py}}(r_y^\star - r_y) - k_{\text{dy}} w_y \\ k_{\text{pz}}(w_z^\star - w_z) \end{bmatrix}, \tag{4.38}$$

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 4.1, were determined experimentally.

4.4.2 Onboard 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. 4.4). 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.

Table 4.1: 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

4.4.2.1 LIDAR Segmentation

In order to provide a more efficient obstacle representation to the collision avoidance algorithm in this chapter, the 2D range data provided by the spinning LIDAR will be segmented through the clustering and split-and-merge algorithm as discussed in [54]. 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. 4.5(a)). 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 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. 4.5(b)). 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. 4.5(c)). 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. 4.5(d)). The first and last point in each of the resulting segments are then considered as vertices of the obstacles for the collision avoidance algorithm.

4.4.2.2 Minkowski Difference Algorithm

The data that have been processed by the LIDAR segmentation, discussed previously, are 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 [55], [56] 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 [56].

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(distance$(L, s_k \in \mathcal{L}_j)$)
  else
    Segment $\mathcal{L}_j$ is complete
  end if
end for

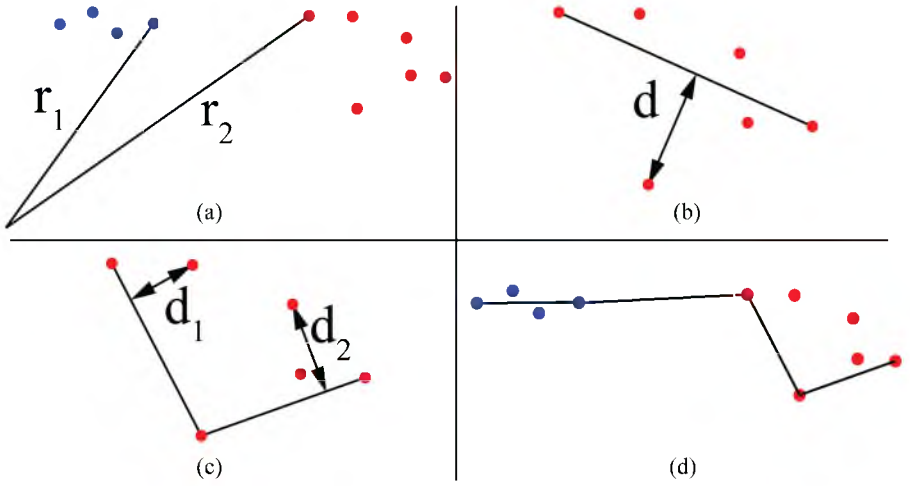

Figure 4.5: Shown is the graphical representation of the clustering and split-and-merge algorithm for the LIDAR data. (a) First, the data are clustered based on the difference in the range data. If two points have a radial distance greater than a threshold $|r_1 - r_2| > r_{\text{thresh}}$ then they 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.

4.4.2.3 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 [23], the algorithm was studied for varying amounts of uncertainty and confidence bounds. Provided the uncertainty covariance matrices, the experimental application in this chapter could follow that approach directly, but 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. (4.24), the algorithm will provide 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 multirotor vehicle in constrained indoor environments is also subject to aerodynamic disturbances which can be large and difficult to model, as shown in current research [57], [58]. These disturbances will also likely violate the assumptions in [23] 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.

4.5 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 were 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 performed as expected, and the measured responses also agreed with simulations and expected behaviors. The results are described next.

In the first case, the quadcopter was flown straight at a wall and the results are shown in Fig. 4.6. The series of images shown in the first two columns in Fig. 4.6 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 chapter 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 $t = 2$ 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 ($t = 4, 6, 8$ s) and a possible collision was detected by the onboard 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 toward 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 ($t = 6, 8$ 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. 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. 4.7. As shown, the robot was first flown toward a wall on its right and then it strafed along that wall into the corner ($t = 2, 4$ s), followed by strafing along the wall in front of the robot ($t = 6, 8, 10$ s), as shown in the sequence of images in the first column in Fig. 4.7. 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. 4.7 that there are times when the LIDAR returns a maximum range reading when it should

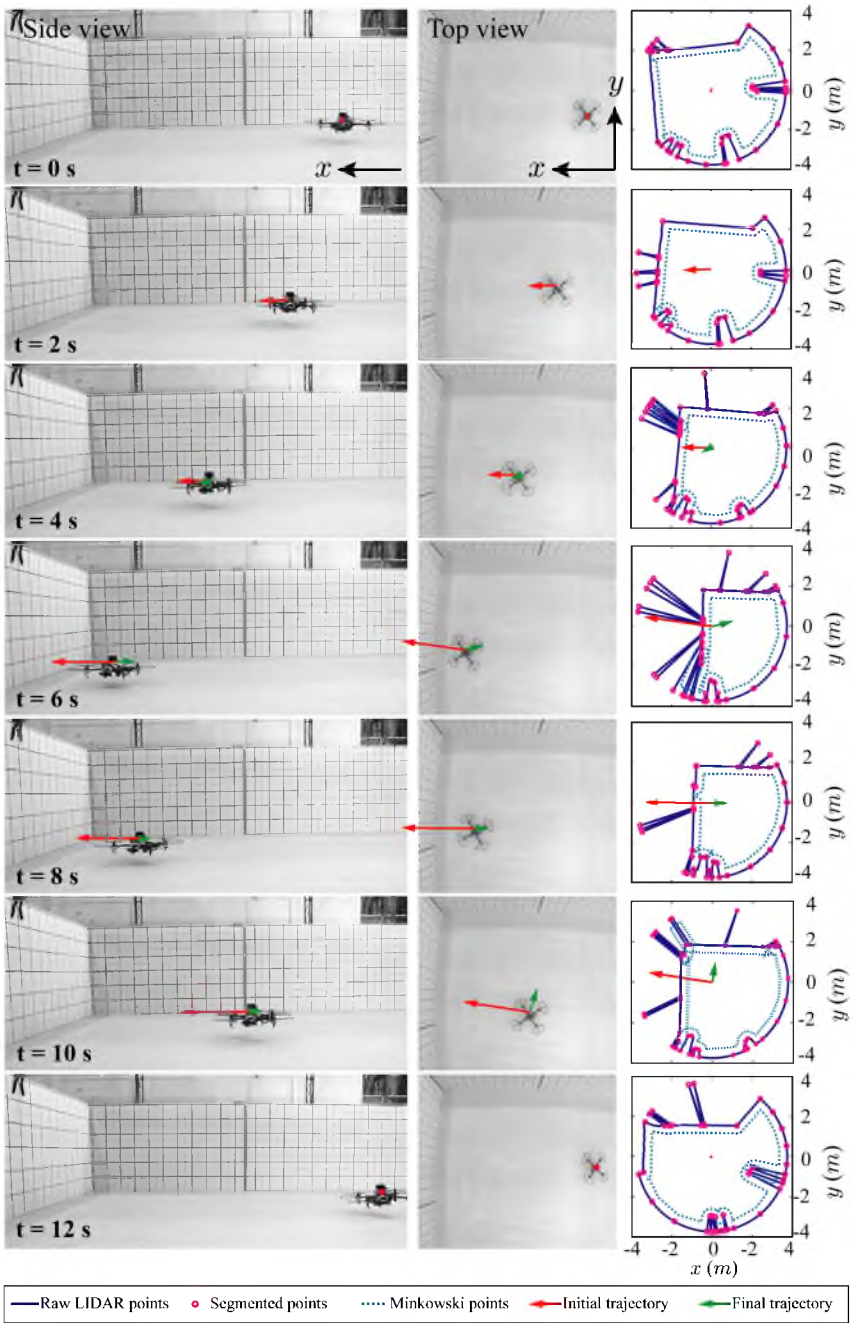

Figure 4.6: Case 1: Experimental results where the aerial robot was flown directly at a wall in front of it. A sequence of time steps is shown along with the data from the sensors and the resultant desired trajectories. The left column shows the side view where the quadcopter moves toward the wall and 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.

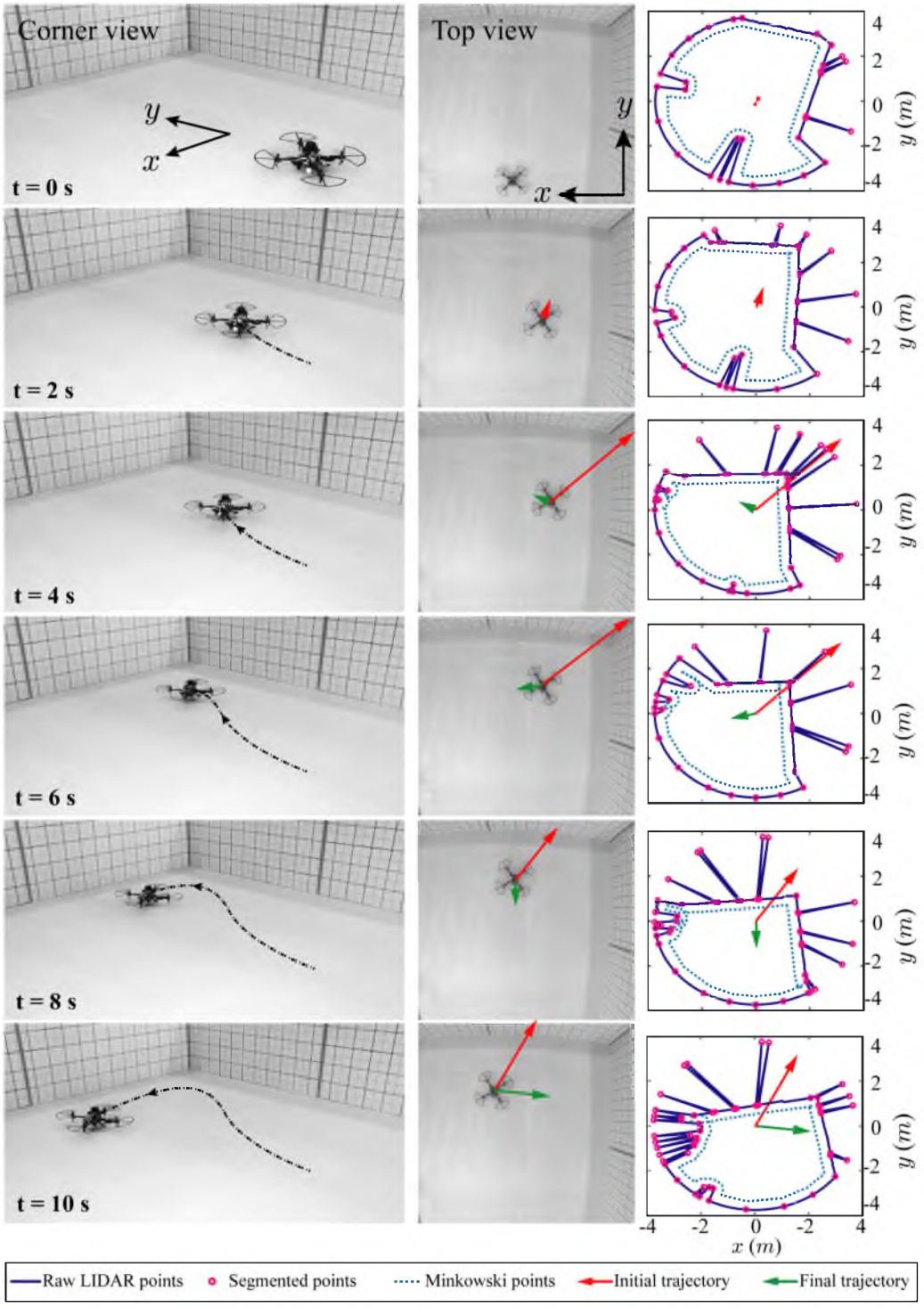

Figure 4.7: Case 2: Experimental results where the aerial robot was flown into a corner. The quadcopter is first flown toward the wall on the right, then strafes along it temporarily ($t = 2, 4$ s) before moving along the wall in front of it ($t = 6, 8, 10$ s), resulting in collision-free motion with respect to both of the walls.

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 its ability to handle internal obstacles (such as office file cabinets). As shown in Fig. 4.8, the quadcopter was flown at a narrow obstacle that is within the environment rather than only using the exterior walls. The quadcopter flies toward the cabinet obstacle ($t = 0, 1$ s) 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 toward the wall and come to a stop ($t = 4 - 9$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 Sec. 4.4.1.1 and the collision avoidance algorithm presented herein. The simulation and experimental results are shown in Fig. 4.9(a) and (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 toward 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 continued to fly down the straight hallway ($t = 14, 16$ s). Both the simulated trajectories and LIDAR scan data showed good agreement with the measured experimental results shown in Fig. 4.9(b). Thus, the results

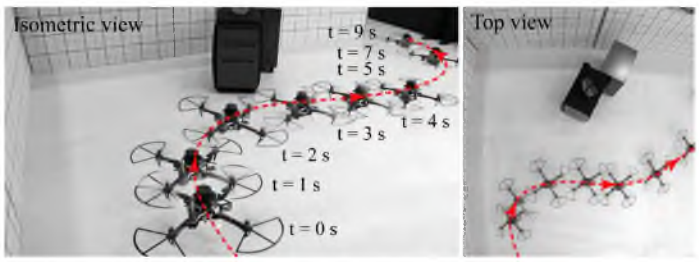

Figure 4.8: Case 3: In this experiment, the quadcopter is flown at a narrow obstacle that is within the environment, rather than only using the horizontal walls. The quadcopter flies toward the cabinet ($t = 0, 1$s) until it then strafes to the right along the front face of the cabinet ($t = 2, 3$s). After passing the cabinet, the quadcopter is able to move toward the wall and come to a stop ($t = 4 - 9$s).

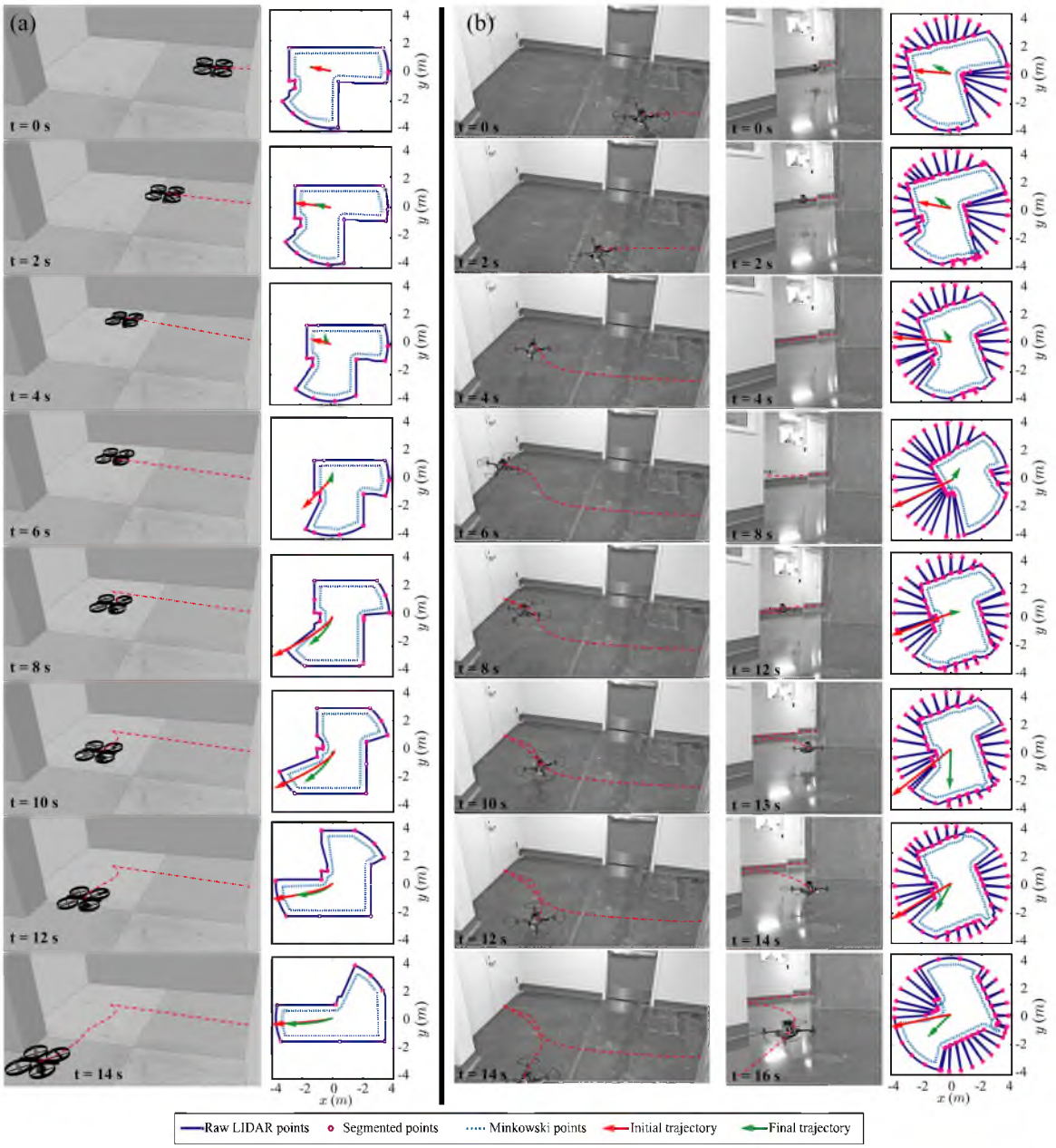

Figure 4.9: Case 4: (a) Simulations and (b) experiments of the robot flying through an "S"-shaped basement hallway. The quadcopter was flown from a straight hallway toward a wall ($t = 0, 2, 4, 6$ s), where it strafes to the left ($t = 8, 10, 12, 13$ s), then the space opened up into a straight hallway and the robot continues to fly down the straight hallway ($t = 14, 16$ s).

in this case show 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.

4.6 Conclusions and Future Work

In this chapter, a feedforward-based automatic collision avoidance algorithm was presented and implemented on an experimental quadcopter with onboard sensing and computation. From a pilot's input, the algorithm predicts the trajectory given the 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 objects in the environment. These range data were 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 onboard 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.

4.7 Acknowledgments

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

4.8 References

[1] A. Abdilla, A. Richards, and S. Burrow, "Power and endurance modelling of battery-powered rotorcraft," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 675 –680.

[2] M. Gatti, F. Giulietti, and M. Turci, "Maximum endurance for battery-powered rotary-wing aircraft," *Aerospace Science and Technology*, vol. 45, pp. 174 –179, 2015.

[3] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: modeling, estimation, and control of quadrotor," *IEEE Robotics & Automation Mag.*, vol. 19, no. 3, pp. 20 –32, 2012.

[4] J. Han, Y. Xu, L. Di, and Y. Chen, "Low-cost multi-UAV technologies for contour mapping of nuclear radiation field," *J. of Intelligent & Robotic Systems*, vol. 70, no. 1, pp. 401 –410, 2013.

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

[6] M. Neri, A. Campi, R. Suffritti, F. Grimaccia, P. Sinogas, O. Guye, C. Papin, T. Michalareas, L. Gazdag, and I. Rakkolainen, "SkyMedia - UAV-based capturing of HD/3D content with WSN augmentation for immersive media experiences," in *IEEE Int. Conf. on Multimedia and Expo*, 2011, pp. 1–6.

[7] S. Waharte and N. Trigoni, *Supporting search and rescue operations with uavs*, Conference Paper, 2010.

[8] A. Barrientos, J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, "Aerial remote sensing in agriculture: a practical approach to area coverage and path planning for fleets of mini aerial robots," *J. of Field Robotics*, vol. 28, no. 5, pp. 667–689, 2011.

[9] G. A. Landis, "Robots and humans: synergy in planetary exploration," *Acta astronautica*, vol. 55, no. 12, pp. 985–990, 2004.

[10] P. S. Lin, L. Hagen, K. Valavanis, and H. Zhou, "Vision of unmanned aerial vehicle (UAV) based traffic management for incidents and emergencies," in *12th World Congress on Intelligent Transport Systems*, 2005, pp. 1–12.

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

[12] H. S. Trammell, A. R. Perry, S. Kumar, P. V. Czipott, B. R. Whitecotton, T. J. McManus, and D. O. Walsh, "Using unmanned aerial vehicle-borne magnetic sensors to detect and locate improvised explosive devices and unexploded ordnance," in *Proc. SPIE Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IV*, vol. 5778, 2005.

[13] T. Brown, S. Doshi, S. Jadhav, and J. Himmelstain, "Test bed for a wireless network on small UAVs," in *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop, and Exhibit*, 2004, pp. 20–23.

[14] H. Yoshimoto, K. Jo, and K. Hori, "Toward entertainment blimps for everyone by everyone," in *Proceedings of the seventh ACM conference on creativity and cognition*, 2009, pp. 445 –446.

[15] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, "Supporting wilderness search and rescue using a cameraequipped mini UAV," *J. of Field Robotics*, vol. 25, no. 1-2, pp. 89 –110, 2008.

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

[17] D. T. Cole, S. Sukkarieh, and A. H. Göktoğan, "System development and demonstration of a uav control architecture for information gathering missions," *Journal of Field Robotics*, vol. 23, no. 6-7, pp. 417–440, 2006.

[18] Z. Cook, L. Zhao, J. Lee, and W. Yim, "Unmanned aerial system for first responders," in *12th Int. Conf. on Ubiquitous Robots and Ambient Intelligence*, 2015, pp. 306 –310.

[19] M. Kumar, K. Cohen, and B. Homchaudhuri, "Cooperative control of multiple uninhabited aerial vehicles for monitoring and fighting wildfires," *Journal of Aerospace Computing, Information, and Communication*, vol. 8, no. 1, pp. 1–16, 2011.

[20] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "UAVs for smart cities: opportunities and challenges," in *Int. Conf. on Unmanned Aircraft Systems*, 2014, pp. 267 –273.

[21] K. P. Valavanis and G. J. Vachtsevanos, "UAV sense, detect and avoid: introduction," in *Handbook of Unmanned Aerial Vehicles*. Springer Netherlands, 2014, pp. 1813 – 1816.

[22] J. Israelsen, M. Beall, D. Bareiss, D. Stuart, E. Keeney, and J. van den Berg, "Automatic collision avoidance for manually tele-operated unmanned aerial vehicles," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 6638–6643.

[23] D. Bareiss, J. van den Berg, and K. K. Leang, "Stochastic automatic collision avoidance for tele-operated unmanned aerial vehicles," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015, pp. 4818–4825.

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

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

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

[27] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

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

[29] S. D'Attanasio, O. Tonet, G. Megali, M. C. Carrozza, and P. Dario, "A semi-automatic handheld mechatronic endoscope with collision-avoidance capabilities," in *IEEE Int. Conf. on Robotics and Automation*, 2000, pp. 1586–1591.

[30] M. Niewenhuisen and S. Behnke, "3d planning and trajectory optimization for real-time generation of smooth MAV trajectories," in *European Conf. on Mobile Robots*, 2015, pp. 1–7.

[31] S. Patil, J. van den Berg, and R. Alterovitz, "Estimating probability of collision for safe planning under gaussian motion and sensing uncertainty," in *IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 3238–3244.

[32] J. Muller and G. S. Sukhatme, "Risk-aware trajectory generation with application to safe quadrotor landing," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 3642–3648.

[33] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart, "Motion- and uncertainty-aware path planning for micro aerial vehicles," *J. of Field Robotics*, vol. 31, no. 4, pp. 676–698, 2014.

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

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

[36] J. Mendes and R. Ventura, "Assisted teleoperation of quadcopters using obstacle avoidance," *J. of Automation, Mobile Robotics, & Intelligent Systems*, vol. 7, no. 1, pp. 54–58, 2013.

[37] C. Brand, M. J. Schuster, H. Hirschmuller, and M. Suppa, "Stereo-vision based obstacle mapping for indoor/outdoor SLAM," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 1846–1853.

[38] F. Rehmtullah and J. Kelly, "Vision-based collision avoidance for personal aerial vehicles using dynamic potential fields," in *12th Conf. on Computer and Robot Vision*, 2015, pp. 297–304.

[39] P. Stegagno, M. Basile, H. H. Bulthoff, and A. Franchi, "A semi-autonomous UAV platform for indoor remote operation with visual and haptic feedback," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 3862–3869.

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

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

[42] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: a factored solution to the simultaneous localization and mapping problem," in *AAAI-02 Proceedings*, 2002, pp. 593–598.

[43] K. Huh, J. Park, J. Hwang, and D. Hong, "A stereo vision-based obstacle detection system in vehicles," *Optics and Lasers in engineering*, vol. 26, no. 2, pp. 168–178, 2008.

[44] N. Bernini, M. Bertozzi, L. Castangia, M. Patander, and M. Sabbatelli, "Real-time obstacle detection using stereo vision for autonomous ground vehicles: a survey," in *IEEE Int. Conf. on intelligent Transportation Systmes*, 2014, pp. 873–878.

[45] P. Agrawal, A. Ratnoo, and D. Ghose, "Vision based obstacle detection and avoidance for UAVs using image segmentation," in *AIAA Guidance, Navigation, and Control Conf.*, 2015, pp. 848–857.

[46] L. Matthies, R. Brockers, Y. Kuwata, and S. Weiss, "Stereo vision-based obstacle avoidance for micro air vehicles using disparity space," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 3242–3249.

[47] S. Saha, A. Natraj, and S. Waharte, "A real-time monocular vision-based frontal obstacle detection and avoidance for low cost UAVs in GPS denied environment," in *IEEE Int. Conf. on Aerospace Elec. and Remote Sensing Tech.*, 2014, pp. 189–195.

[48] L. Mejias, S. McNamara, J. Lai, and J. Ford, "Vision-based detection and tracking of aerial targets for UAV collision avoidance," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010, pp. 87–92.

[49] O. Astilla, J. Guerrero, R. Mendoz, and M. Teriz P. Roxas, "Obstacle avoidance of hybrid mobile-quadrotor vehicle with range sensors using fuzzy logic control," in *Int. Conf. on Humanoid, Nanotechnology, Inf. Tech., Communication and Control, Env. and Management*, 2015, pp. 1–8.

[50] D. Maier, A. Hornung, and M. Bennewitz, "Real-time navigation in 3d environments based on depth camera data," in *Int. Conf. on Humanoid Robots*, 2012, pp. 692–697.

[51] T. Wang, L. Bu, and Z. Huang, "A new method for obstacle detection based on kinect depth image," in *Chinese Automation Congress*, 2015, pp. 537–541.

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

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

[54] V. Nguyen, S. Gachter, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2d range data for indoor mobile robotics," *Autonomous Robots*, vol. 23, no. 2, pp. 97–111, 2007.

[55] J.-M. Lien, "Point-based minkowski sum boundary," in *15th Pacific Conf. on Computer Graphics and Applications*, 2007, pp. 261–270.

[56] E. Behar and J.-M. Lien, "Fast and robust 2d minkowski sum using reduced convolution," in *IEEE/RSJ Int. Conf. on Int. Robots and Systems*, 2011, pp. 1573–1578.

[57] C. G. Hooi, F. D. Lagor, and D. A. Paley, "Flow sensing, estimation and control for rotorcraft in ground effect," in *Proc. IEEE Aerospace Conference*, 2015, pp. 1 –8.

[58] D. W. Yeo, N. Sydney, and D. A. Paley, "Onboard flow sensing for rotary-wing uav pitch control in wind," in *AIAA Guidance, Navigation, and Control Conf.*, 2016, pp. 1386–1396.

CHAPTER 5

STUDY OF IMPROVED PILOT PERFORMANCE USING AUTOMATIC COLLISION AVOIDANCE FOR TELE-OPERATED UNMANNED AERIAL VEHICLES

5.1 Introduction

Unmanned aerial vehicles (UAVs), particularly small, low-cost platforms, have gained considerable attention for civil and commercial applications ranging from mapping [1] to precision farming [2], traffic management [3], and environmental monitoring [4]. More recently, the emergence of small multirotor UAVs (such as quadcopters), which can access indoor locations and maneuver through environments that are hard to reach or unsafe for humans, has captured the attention of the public-safety sector and law-enforcement officials as a viable tool to enhance situational awareness for search and rescue, law enforcement, and/or emergency response [5]–[7]. However, one of the most daunting tasks for even a skilled UAV pilot is controlling the aircraft for collision avoidance, especially in tight and compact environments such as inside of a partially collapsed building where usually the only feedback information is a live-camera feed through first-person view (FPV) mode (see Fig. 5.1 illustrating the typical UAV system that is controlled through a remote command station). Thus, automatic collision avoidance technology for tele-operated UAVs (as well as mobile ground robots) is critical and necessary to allow pilots to focus on higher-priority tasks such as locating survivors and acting quickly to help assist survivors or call for additional support.

To quantitatively investigate the impact of automatic collision avoidance technology on UAV-pilot performance, the contribution of this paper is a human-subject study that compares the performance between a feedforward-based collision avoidance algorithm [8], [9], a

This material is based upon work supported by the National Science Foundation, Partnership for Innovation Program, Grant No. 1430328.

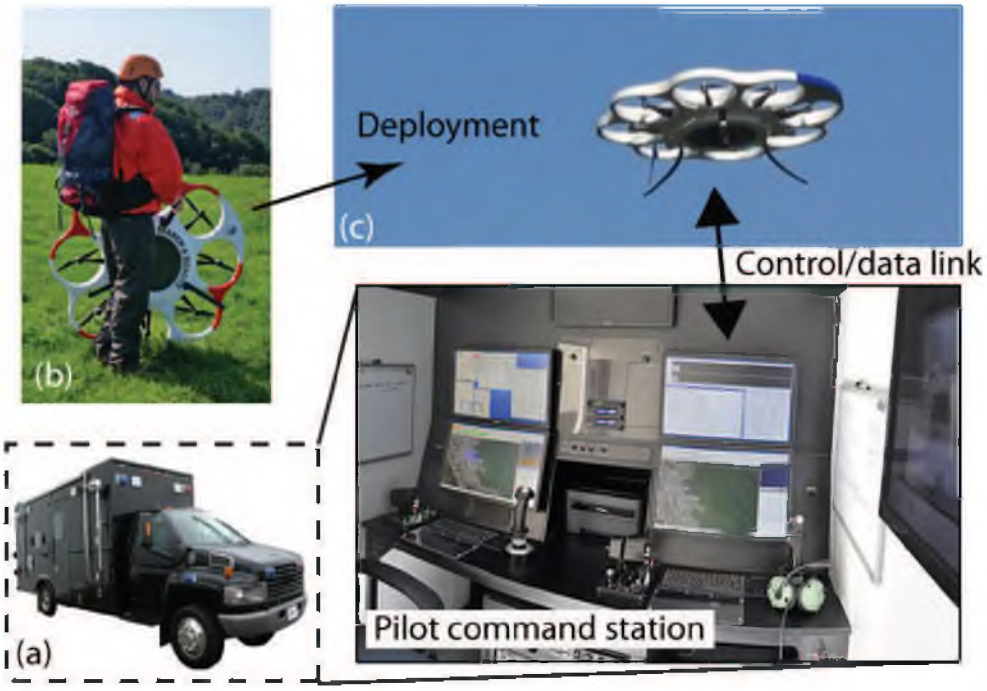

Figure 5.1: A UAV system for search and rescue and emergency response, where pilots control the unmanned aerial vehicle (UAV) through (a) a mobile command station or similar interface following (b) deployment of (c) the UAV with on-board cameras and sensors. Control signals and data flow between the UAV and command station. Images courtesy of Mike Richards and Drone America, Inc.

basic risk (potential) field algorithm [10], and full manual control. Specifically, experiments are described where pilots operate a simulated UAV system running the algorithms through three maze-like environments. In the experiments, the number of collisions, the path length, trial time, and average speed are recorded. There are four hypotheses being tested in this paper. First, it is hypothesized that the algorithm in this paper will result in fewer collisions than manual control. Second, of the trials that do not collide, it is hypothesized that there will be higher operating speeds (i.e., shorter completion times) with the algorithm in this paper over manual control. Third, it is hypothesized that this algorithm will result in fewer collisions than the potential-field variant, the basic risk field algorithn [10]. Lastly, it is hypothesized that the algorithm in this paper will provide higher operating speeds than the basic risk field algorithm. The first, second, and fourth hypotheses are supported by the experiments, while the third hypothesis is inconclusive, but suggests that there is not a significant difference in the frequency of collisions between the two algorithms.

5.2 State-of-the-Art in Collision Avoidance

Early research on motion planning and collision avoidance for mobile robots included potential-field planners [11] and the vector field histogram (VFH) approach [12]. Improvements were made to the VFH in [13]. Although these algorithms are effective, they do have some potential limitations for applications such as search and rescue. For instance, these algorithms have an inherent requirement to keep the robot some minimum distance away from the obstacles in the environment. The need to maintain a minimum distance from the walls comes from the fact that the algorithms do not explicitly consider the dynamics of the robot. However, in a search and rescue scenario the robot may need to be controlled near walls in constrained environments or in order to more quickly to survey the environment. Thus, factoring in the robot's dynamics can improve the performance of the collision avoidance process.

Collision avoidance methods that do consider the robot's dynamics typically require a global knowledge of the environment [14]–[16]. Unfortunately, such information may not be readily available at the time of search and rescue and is often not practical. Furthermore, these algorithms are more computationally expensive than the reactive planners such as potential-fields and VFH.

Herein, a feedforward-based local collision avoidance algorithm is presented that has similarities to both classes of collision avoidance algorithms [8], [9]. More specifically, the algorithm considers the robot's dynamics and extrapolates the robot's trajectory given an operator's input. The resulting trajectory is checked for collisions against the obstacles in the environment. If a collision is predicted, the user's input is modified to guide the robot along a collision-free trajectory. Similar to the reactive planners, such as potential-field, this algorithm only requires a limited knowledge of the local environment in the immediate vicinity of the robot. It also has similarities to more complex planners through the propagation of the trajectory using the robot's dynamics. However, rather than optimizing this trajectory explicitly, the algorithm is designed to alter the user's input directly which results in collision-free motion while maintaining the user's intent as closely as possible.

The remainder of this paper is structured as follows. The feedforward-based automatic collision avoidance algorithm is reviewed in Sec. 5.3. The methodology of the experiments is presented in Sec. 5.5 and the results are presented and discussed in Sec. 5.6. Finally, concluding remarks and a discussion of future work are presented in Sec. 5.8.

5.3 Automatic Collision Avoidance

This section provides a review of the feedforward-based automatic collision avoidance (ACA) algorithm studied in this paper. The full theoretical details for the deterministic and stochastic approaches are presented in [8] and [9], respectively.

5.3.1 System Equations and Robot Workspace

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

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{5.1}$$

where $\mathbf{x}(t) \in \mathcal{X}$ and $\mathbf{u}(t) \in \mathcal{U}$ are the state and control input at time t, respectively.

Given an initial state $\mathbf{x} = \mathbf{x}(0)$ and a constant control input $\mathbf{u}$ up to the time-horizon τ, the state of the robot for $t > 0$ is defined by

$$\mathbf{x}(t) = \mathbf{g}(\mathbf{x}, \mathbf{u}, t), \tag{5.2}$$

where $\mathbf{g} \in \mathcal{X} \times \mathcal{U} \times \mathbb{R} \to \mathcal{X}$ represents the solution to the differential equation (5.1).

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.

Remark: *In order to maintain compatibility with the implementation of on-board sensing, those regions of the workspace that are occluded by the obstacles as seen from the current state of the robot are also considered obstacles. In other words, the subspace of the workspace that cannot be seen by the robot is also an obstacle.*

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

The model has a 12-dimensional state $\mathbf{x} = [\mathbf{p}^T, \mathbf{v}^T, \mathbf{r}, \mathbf{w}]^T \in \mathcal{X}$ that consists of position $\mathbf{p} \in \mathbb{R}^3$, velocity $\mathbf{v} \in \mathbb{R}^3$, Euler angles $\mathbf{r} \in \mathbb{R}^3$, and angular velocity $\mathbf{w} \in \mathbb{R}^3$. The 4-dimensional control input $\mathbf{u} = [r_x^\star, r_y^\star, v_z^\star, w_z^\star]^T \in \mathcal{U}$ consists of the desired roll and pitch angles (roll, pitch, and yaw), $r_x^\star$ and $r_y^\star$, respectively, the desired vertical velocity $v_z^\star$, and the desired yaw rate $w_z^\star$. Assuming a quadcopter UAV system, the equations of motion are given as

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{5.3}$$

$$\dot{\mathbf{v}} = R\left[0, 0, k_{\text{pv}}(v_z^\star - v_z)\right]^T - \mathbf{g} - k_{\text{drag}}\mathbf{v}, \tag{5.4}$$

$$\dot{\mathbf{r}} = \mathbf{w}, \tag{5.5}$$

$$\dot{\mathbf{w}} = \begin{bmatrix} k_{\text{px}}(r_x^\star - r_x) - k_{\text{dx}} w_x \\ k_{\text{py}}(r_y^\star - r_y) - k_{\text{dy}} w_y \\ k_{\text{pz}}(w_z^\star - w_z) \end{bmatrix} \tag{5.6}$$

where R is the rotation from the quadcopter body frame into the world frame, the terms $k_{\text{pv}}, k_{\text{px}}, k_{\text{dx}}, k_{\text{py}}, k_{\text{dy}}, k_{\text{pz}}$, and k_{drag} are gains whose values (given in Table 5.1) are equal to an analogous physical system. Similar to many physical quadcopters, the simulated quadcopter has a maximum limit on the roll and pitch angles that is 0.35 rad (approximately 20.0 degrees).

5.3.2 Problem Statement

The collision avoidance problem is defined as finding a minimal change $\Delta\mathbf{u} \in \mathcal{U}$ to the control input $\mathbf{u} \in \mathcal{U}$ given the initial state $\mathbf{x} \in \mathcal{X}$ of the robot to avoid collisions with obstacles within a time horizon τ, hence

$$\begin{aligned} &\text{minimize: } \Delta\mathbf{u}^T Q \Delta\mathbf{u} \\ &\text{subject to: } \forall t \in [0, \tau] :: \mathcal{R}(\mathbf{g}(\mathbf{x}, \mathbf{u} + \Delta\mathbf{u}, t)) \cap \mathcal{O} = \emptyset, \end{aligned} \tag{5.7}$$

where $Q \in \mathbb{R}^{n \times n}$ is a positive-definite weighting matrix.

5.3.3 Approach

The details of the automatic collision avoidance algorithm are presented below, where additional details are found in [9].

Given the robot's current state $\mathbf{x}$ and the current control input $\mathbf{u}$ (from the operator), the positions of the robot in the future are found by

$$\mathbf{p}(t, \Delta\mathbf{u}) \approx \mathbf{p}^\star(t) + J(t)\Delta\mathbf{u}, \tag{5.8}$$

where $\mathbf{p}^\star(t)$ is the position the robot would obtain if the operator's input remains constant, i.e., $\Delta\mathbf{u} = \mathbf{0}$, and $J(t)$ is the Jacobian of the position with respect to the input.

For a trajectory that is determined to be collision free ($\forall t \in [0, \tau] :: \mathcal{R}(\mathbf{p}^\star(t)) \cap \mathcal{O} = \emptyset$), the operator's current input $\mathbf{u}$ is deemed safe and does not need to be changed, hence the change in input is set to zero: $\Delta\mathbf{u} = \mathbf{0}$. Conversely, if a collision does occur ($\forall t \in [0, \tau] :: \mathcal{R}(\mathbf{p}^\star(t)) \cap \mathcal{O} \neq \emptyset$) the operator's input leads to a collision and must be corrected in order for the robot to obtain a collision-free trajectory, hence the change in input is nonzero: $\Delta\mathbf{u} \neq \mathbf{0}$. The process to select a nonzero change in input is discussed next.

Table 5.1: Quadcopter Model Parameters

Parameter	k_{pv}	k_{px}	k_{dx}	k_{py}	k_{dy}	k_{pz}	k_{drag}
Value	10.0	150.0	2.5	150.0	2.5	3.5	0.25

Let $\mathbf{p}_c$ be the first point along the trajectory in which the robot collides with an obstacle (see Fig. 5.2), thus

$$\mathbf{p}_c = \mathbf{p}^\star(\min\{t \in [0, \tau] \mid \mathcal{R}(\mathbf{p}^\star(t)) \cap \mathcal{O} = \emptyset\}). \tag{5.9}$$

Given a unit normal vector $\mathbf{n}$ of the obstacle $\mathcal{O}$ that points into the free workspace, consider a halfspace with the same normal $\mathbf{n}$ (pointing toward the free space) that provides a convex approximation of the local free space. The halfspace is located at the collision point $\mathbf{p}_c$, determined by Eq. (5.9).

Given Eq. (5.9), a linear constraint is defined on the position $\mathbf{p}(\tau, \Delta\mathbf{u})$ of the robot at time τ,

$$\mathbf{n}^T\mathbf{p}(\tau, \Delta\mathbf{u}) > \mathbf{n}^T\mathbf{p}_c. \tag{5.10}$$

The constraint on the robot's position in Eq. (5.10) can be transformed into a constraint on its change in input $\Delta\mathbf{u}$ by substituting in Eq. (5.8), thence

$$\mathbf{n}^T J(\tau)\Delta\mathbf{u} > \mathbf{n}^T(\mathbf{p}_c - \mathbf{p}^\star(\tau)). \tag{5.11}$$

Equation (5.7) is approximated using Eq. (5.11) as

$$\begin{aligned} &\text{minimize: } \Delta\mathbf{u}^T Q \Delta\mathbf{u} \\ &\text{subject to: } \mathbf{n}^T J(\tau)\Delta\mathbf{u} > \mathbf{n}^T(\mathbf{p}_c - \mathbf{p}^\star(\tau)), \end{aligned} \tag{5.12}$$

where solving this convex optimization, such as is done by the RVO library in [17], provides a collision-free change in input $\Delta\mathbf{u}$, where finally the total control input provided to the robot is $\mathbf{u} + \Delta\mathbf{u}$.

5.3.4 Iteration for Convex Corners and Edges

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 $t \in [0, \tau]$. In particular, this is true near convex edges or corners of the workspace as shown in Fig. 5.2. However, the approach can simply be repeated in an iterative fashion to solve this problem, as described in [8].

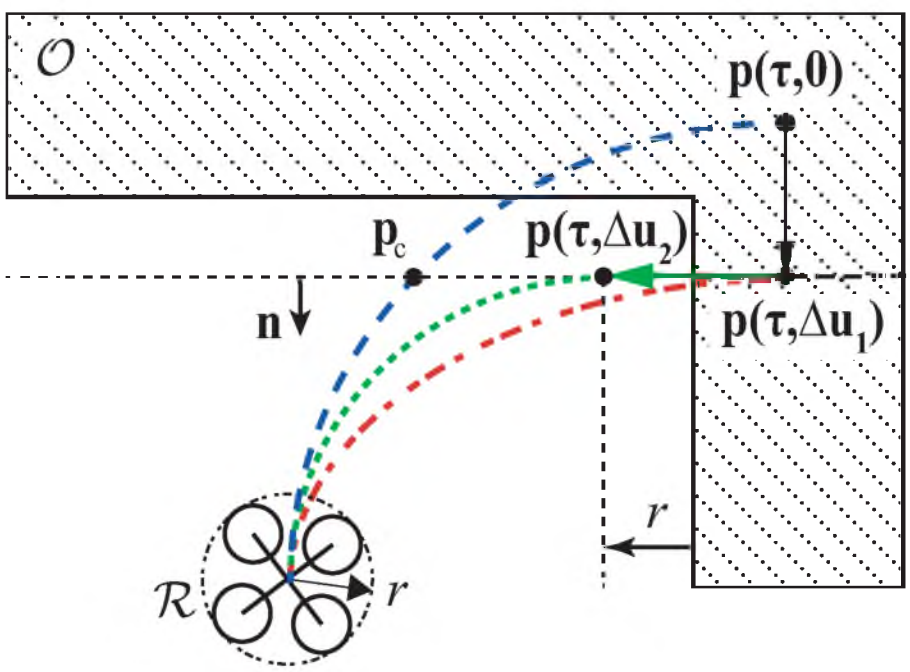

Figure 5.2: Robot with its bounding geometry $\mathcal{R}$, where the estimated trajectory for a user's input is given with the desired position at the time horizon $\mathbf{p}(\tau, 0)$ causing a collision with the obstacle $\mathcal{O}$ at $\mathbf{p}_c$. The new collision-free input at time τ is solved for; however, in convex corners this process needs to be iterated to detect possible collisions between the updated trajectory. Given the first iteration, the new estimated trajectory with the desired position as $\mathbf{p}(\tau, \Delta\mathbf{u}_1)$ and a new input is chosen if a collision occurs. This is repeated until the trajectory is collision free, as shown by $\mathbf{p}(\tau, \Delta\mathbf{u}_2)$.

5.3.5 Including Yaw as an Additional Degree-of-Freedom

In [8], [9], yaw motion is treated as a redundant degree of freedom and is held constant. However, to better enable a pilot to survey an area in a search-and-rescue scenario, holding the yaw constant limits performance. Yaw motion is especially necessary if the pilot is flying through a first-person video feed using a forward-facing camera.

To enable the yaw to be controlled by the pilot, the yaw degree of freedom is not affected by the algorithm. The algorithm determines the feedforward trajectory estimate assuming the user's current commanded yaw rate remains constant over the time horizon τ, similar to the desired roll and pitch angles. Through the Jacobian in Eq. (5.8), the algorithm can determine new roll and pitch angles and vertical velocity to avoid a collision given a constant yaw rate.

5.4 Potential-Field for UAV Tele-operation

The potential-field algorithm provides a repulsive force on the robot based on the distance between the robot and an obstacle detected by a range sensor [11]. The repulsive

force increases as distance between the robot and the obstacle decreases. However, since this algorithm produces forces only based on the distance, it can still provide large repulsive forces even if the robot is moving *away* from a nearby obstacle, which is not desirable for tele-operated applications.

In [18], the potential-field algorithm was augmented to include the velocity and acceleration constraints of a robot. However, in this approach the repulsive force is zero when the robot has no velocity component towards the obstacle. This is undesirable for a tele-operation application where a user may suddenly provide a control input towards that obstacle and the robot can collide if a repulsive force cannot counter that input fast enough. Lam *et al.* [10] presented the basic risk field (BRF), a potential-field variant, to address this concern. Their approach provides a small repulsive force for nearby obstacles even when there is no velocity component of the robot towards that obstacle. Therefore, there is always a small repulsion force pushing the robot away from obstacles, but the repulsive force is only a large force when reacting to velocities toward the given obstacle. The potential function $P(d, v_i)$ that defines these repulsive forces given in [10] is

$$t_{res}(d, v_i) = \frac{a_{max} v_i}{2 d a_{max} - v_i^2}, \tag{5.13}$$

$$P(d, v_i) = \begin{cases} 1, & \text{if } t_{res} \leq 0 \\ 1, & \frac{1}{t_{res}(d,v_i)} + \frac{1}{d} \geq \frac{1}{G} \\ G\left(\frac{1}{t_{res}(d,v_i)} + \frac{1}{d}\right), & \text{otherwise,} \end{cases}$$

where a_{max} is the maximum deceleration of robot toward an obstacle, v_i is the robot's velocity component toward the obstacle, d is the distance between the robot and obstacle, and G is a gain to tune the magnitude of the repulsive gain.

5.5 Experimental Methods

5.5.1 Subjects

Three experiments were performed by 24 subjects (eight per experiment). The subjects were recruited from the University of Utah student population. The subjects had the physical ability to use a commercial video game console controller and were at least 18 years of age. The subjects were not compensated for their participation.

The experiment was approved by the University of Utah Institutional Review Board.

5.5.2 Device

The quadcopter model provided in Sec. 5.3.1 is implemented in simulation on a desktop computer with an Intel Core i5-3470 3.2 GHz processor, 8GB RAM, and 64-bit Ubuntu 12.04 operating system. The algorithms are implemented using the Robot Operating System (ROS) [19]. The aircraft is flown in first-person view (FPV) mode, where simulated FPV is made available to pilots operating the UAV. The simulator includes a 2D LIDAR to detect the obstacles in the environment in real time during the experiments.

Three environments were created in simulation using the V-REP software package [20] as shown in Fig. 5.3. The environments each used the same starting position of the quadcopter but have different finish locations. The corridors of each environment are either 2 m, 1.5 m, or 1 m wide. The simulated quadcopter has a diameter of 0.564 m.

5.5.3 Design

A full-factorial repeated-measures design is used for the three experiments. There are two factors being considered in the experiments: the control method (manual, automatic collision avoidance, or basic risk field) and the environment (mazes shown in Fig. 5.3). A block design is used in which the three environments are presented to the participant in eight blocks of three, for 24 total trials. The order of the mazes in each block of three mazes is a random permutation. Each environment is seen an equal number of times by all participants.

The experiments compare pairs of control methods. The first and second experiments compare manual control to the ACA algorithm. These studies test the hypotheses that the ACA algorithm will result in fewer collisions than manual control, and that when collisions do not occur, the ACA algorithm will enable higher operating speeds. During a pilot study, it was observed that many subjects preferred to fly the quadcopter similar to a car, where they provided a constant forward input and steered the UAV through yaw. However, the yaw rate input being applied with maximum roll and/or pitch can lead to collisions due to the assumptions in the ACA algorithm's development. Thus, the first experiment allowed the quadcopter to yaw, which led to a relatively high number of collisions. The second experiment is designed such that the quadcopter cannot yaw. Instead, the camera rotates and the pilot's roll and pitch commands are defined in the camera frame and mapped into the robot frame. The third experiment compares the BRF algorithm to the ACA algorithm. This experiment tests the hypotheses that the ACA algorithm will result in fewer collisions than the BRF algorithm, and that the ACA algorithm will enable higher operating speeds than the BRF algorithm. In [10], the simulated quadcopter was a velocity-controlled robot

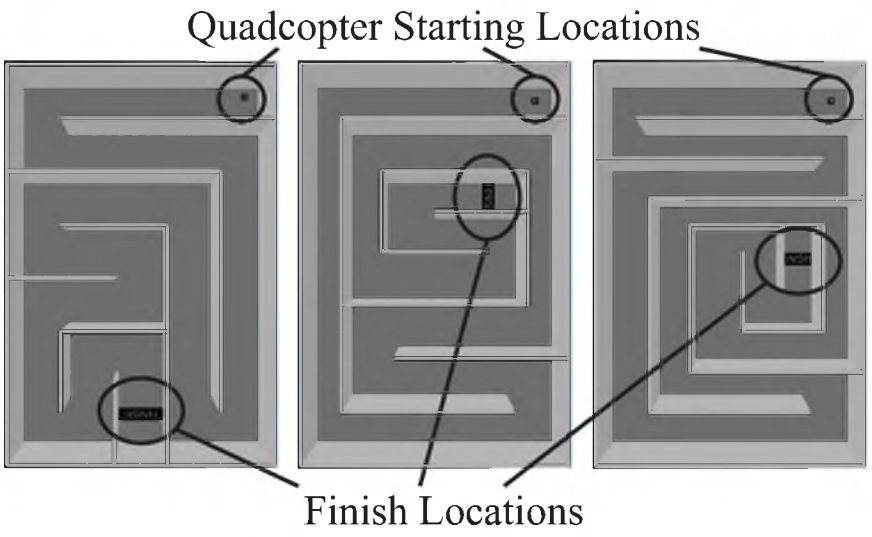

Figure 5.3: Three mazes used during the experimental trials. Every maze has the same starting location with different finish locations as annotated in the image.

with simplified dynamics. The simulations in this paper, however, utilize the full nonlinear dynamics of the quadcopter with inputs including roll and pitch, i.e., accelerations. There were unnecessary oscillations observed when using the BRF so a damping term was included for the roll and pitch inputs.

In each experiment, each participant completed half of their trials (12 trials) using one of the two control methods and then the second half with the alternate control method. The experiments alternated the order of the control methods for each successive participant to attempt to minimize learning effects on the results. For example, the first subject would be tested first using manual control and second with the ACA algorithm, then the second subject would first use the ACA algorithm and use manual control second.

5.5.4 Procedure

For each experiment, the subjects completed two sessions with at least 24 hrs between each session. For each session, the subject sat at a desk and held a wireless game console controller while directly facing a 24in. desktop computer monitor located approximately 24 in. away from the subject. The subject is instructed that, for each trial, they should attempt to complete each maze as fast as possible while avoiding collisions. A collision is indicated by the screen turning red and the current trial stopping automatically.

Before each session, the subject is required to practice with the control method of that session for three minutes. The quadcopter model is the same during the practice as it is during the experiments. The environment during the three minutes of practice consists of

2 m-wide hallways and has a similar appearance to the experiment's environments.

During the experiment, the subject knows which control method they are using but not the technical details of the algorithm (or lack thereof). A verbal cue is given to the subject before each trial begins. A trial ends automatically with a collision or the crossing of the finish line, after which the process is repeated for all 12 trials of that session. Each session typically lasts 15–30 min, for each subject, with each subject completing two sessions.

In the first experiment, the subjects are informed that the automatic collision avoidance algorithm could still have collisions if a full roll or pitch input is applied at the same time as a yaw input, but no further information about the algorithm is provided. In the second and third experiments, when the quadcopter cannot yaw, the subjects are not provided with this additional information about the algorithm.

5.5.5 Measures

To quantify the performance of the subjects operating the simulated quadcopter, their performance is defined by several metrics: if a collision occurred (yes/no), the time to complete each trial, the path length traveled, and the average operating speed. The time, path length, and average operating speed are recorded for the duration of each trial that ended in either a collision or crossing the finish line.

The collision data can be represented as a binomial distribution and analyzed using the Friedman test [21]. The maps had a small effect on the results and are distributed equally in the experiment design, therefore the collisions are only categorized on one level, by which collision avoidance algorithm (or lack thereof) is being used by the subject. The remaining measures can be analyzed using a two-way ANOVA [22].

5.6 Experimental Results

The experimental results (the means, standard deviations, and comparison metrics) are summarized in Table 5.2. Analysis of operating speeds for *only* the trials that are completed (i.e., no collisions) is provided in Table 5.3. Table 5.4 shows if each participant's individual results supported (✓) or contradicted (✗) the hypothesis being tested, with statistical significance at 95% confidence. A "-" represents that no conclusions can be drawn for or against the hypothesis with statistical significance. Figure 5.4 provides box plots of the data set for measures in which ANOVA is performed (i.e., all measures except collisions). For hypothesis two, that the ACA algorithm enables higher operation speeds than manual control for *completed* trials, a value of "N/A" means that no manual trials

Table 5.2: Distribution Statistics for All Subjects and All Trials

(a) Sample Means (μ) and Standard Deviations (s)

		Collisions		Time		Path Len.		Avg. Speed	
		μ	s	μ	s	μ	s	μ	s
E1	Man.	0.89	0.32	66	74	32	25	0.64	0.27
	ACA	0.53	0.501	74	56	54	25	0.87	0.27
E2	Man.	0.86	0.34	42	38	29	23	0.83	0.35
	ACA	0.10	0.31	76	21	71	11	0.97	0.12
E3	BRF	0.073	0.26	112	34	76	17	0.70	0.092
	ACA	0.10	0.31	77	17	73	10	0.96	0.12

(b) Comparison Metrics

	Collisions		Time		Path Len.		Avg. Speed	
	Chi-Sq.	p	F	p	F	p	F	p
E1	33	1.2e-8	1.5	0.23	43	1.1e-9	55	1.6e-11
E2	103	3.7e-24	68	1.9e-13	351	1.7e-38	15	1.8e-4
E3	0.58	0.44	136	6.7e-22	4.1	0.045	468	1.4e-44

Table 5.3: Distribution Statistics for Completed Trials Only, for All Subjects

(a) Sample Means (μ) and Standard Deviations (s)

		Avg. Speed	
		μ	s
E1	Man.	0.46	0.14
	ACA	0.78	0.23
E2	Man.	0.76	0.18
	ACA	0.96	0.12
E3	BRF.	0.70	0.09
	ACA	0.95	0.11

(b) Comparison Metrics

	Avg. Speed	
	F	p
E1	21	2.5e-5
E2	26	1.7e-6
E3	268	5.1e-37

Table 5.4: Hypothesis Results for Individual Subjects

(a) Exp. 1

	H1	H2
S1	✓	N/A
S2	-	-
S3	-	N/A
S4	-	✓
S5	✓	N/A
S6	✓	N/A
S7	✓	✓
S8	-	-

(b) Exp. 2

	H1	H2
S1	✓	✓
S2	✓	N/A
S3	✓	N/A
S4	✓	N/A
S5	✓	✓
S6	✓	✓
S7	✓	✓
S8	✓	-

(c) Exp. 3

	H3	H4
S1	-	✓
S2	-	✓
S3	-	✓
S4	-	✓
S5	-	✓
S6	-	✓
S7	-	✓
S8	-	✓

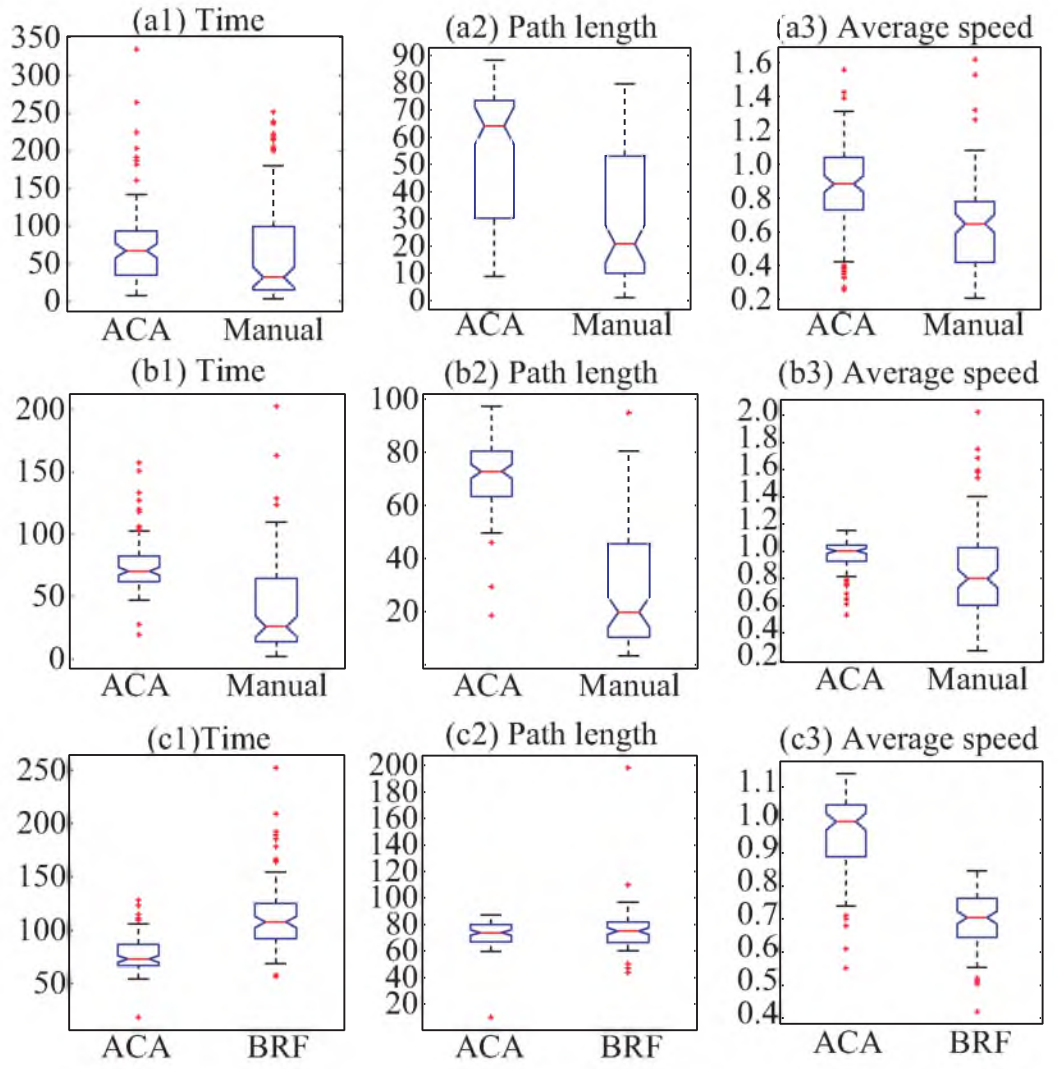

Figure 5.4: Box plots for the experiments comparing automatic collision avoidance (ACA) algorithm, manual control (Manual), and basic risk field (BRF) algorithm. Results in (a1)–(a3) show ACA versus manual control of the UAV with yaw. Results in (b1)–(b3) show ACA versus manual control of the UAV without yaw. Results in (c1)–(c3) show performance of ACA versus BRF algorithm. Left column shows the trial time [(a1)–(c1)], middle column shows the path length [(a2)–(c2)], and right column shows the average speed [(a3)–(c3)].

were completed, or in other words *all* 12 trials resulted in a collision for that participant and no statistical testing can be completed regarding the second hypothesis.

5.6.1 Experiment One: Automatic Collision Avoidance (ACA) vs. Manual Control With Yaw

In regards to Hypothesis 1, the statistical results that include the data from all subjects and all trials (Table 5.2) indicate that the ACA algorithm results in significantly fewer collisions than manual control. The mean number of collisions decreased by 40% from

manual control when using the ACA algorithm. When considering the results for individual subjects (Table 5.4(a)), four out of the eight subjects showed a significant improvement when using ACA, and the other four subjects are inconclusive. No subject showed significant improvement when using manual control.

In regard to Hypothesis 2, the statistical results from all subjects and *completed* trials (Table 5.3) indicate that the ACA algorithm enabled pilots to fly the UAV with higher average operating speeds compared to manual control with a 70% increase in speed. Considering only the individual results (Table 5.4(a)), two of the eight subjects showed improvement in their operating speeds with statistical significance. Another two subjects are inconclusive, with no statistical significance between their speeds. The remaining four subjects did not complete a single trial with manual control. Although the statistics of their operating speeds cannot be assessed, the fact that they did not complete a single trial manually demonstrates the usefulness of the ACA algorithm. No subject showed improvement when using manual control compared to the ACA algorithm.

5.6.2 Experiment Two: ACA vs. Manual Control Without Yaw

The second experiment addressed Hypotheses 1 and 2 as well, but, unlike the first experiment, in this experiment the quadcopter cannot yaw. Hypothesis 1 is strongly supported by the statistical results for all subjects and all trials (Table 5.2). There is an observed 88% decrease in the number of collisions from manual control when using the ACA algorithm in this experiment. Looking at the individual subject's results in Table 5.4(b) shows that, in fact, *every* subject had fewer collisions with the ACA algorithm than they did with manual control with statistical significance.

Considering the statistical results from the completed trials for all subjects (Table 5.3), Hypothesis 2 is supported as well, where the subjects had a higher mean average operating speed with statistical significance with a 26% increase in speed. The individual results in Table 5.4(b) support Hypothesis 2 as well. Four of the eight subjects showed improved operating speeds with statistical significance. One of the eight subjects is inconclusive and the remaining three could not be analyzed because they did not complete a single trial using manual control. None of these subjects showed improvement with significance when using manual control.

5.6.3 Experment Three: ACA vs. Basic Risk Field (BRF)

This experiment addressed Hypotheses 3 and 4. Regarding Hypothesis 3, there is no conclusive evidence found from the experiment. There is no significant difference in the

mean number of collisions between the ACA algorithm and the BRF algorithm. This is the case for all subjects and all trials (Table 5.2) as well as each individual subject (Table 5.4).

Although Hypothesis 3 is inconclusive, the results of this experiment strongly support Hypothesis 4. The subjects performed at higher operating speeds with the ACA algorithm compared to the BRF algorithm as seen by the statistics for all subjects and all trials (Table 5.2) and the completed trials only (Table 5.3). When looking at the individual subjects' results in Table 5.4(c), the hypothesis is supported by *all* eight subjects independently.

5.7 Discussion

It is observed that there is a large difference in the speed increase between the first and second experiment, both comparing ACA to manual control. It is hypothesized that this resulted from the fact that the subjects were informed during the first experiment that a large yaw input at the same time as roll and pitch could result in collision with the ACA algorithm and tended to fly differently in both the manual and ACA trials, typically flying a short distance and stopping and then rotating in place.

The third hypothesis is that there would be fewer collisions using the ACA algorithm than the BRF algorithm. The results of the third experiment are inconclusive with regard to this hypothesis. It was expected that the BRF would perform well regarding collisions due to the potential field's conservative nature, leading to the slower operation speed, so these results are not surprising. The ACA algorithm could be made more conservative to reduce its number of collisions as well. Given the mean number of collisions from the third experiment, obtaining statistical significance would require a much higher power of the study through an increased number of subjects. However, the effect size is small (0.12 when considering the BRF as the control group) and a study with higher power is not likely necessary from a practical standpoint.

The third experiment supported the fourth hypothesis, which is that the ACA algorithm would perform at higher average operating speeds than the BRF algorithm. An increase in speed of approximately 37% is observed in the experiment. This improvement in performance is predicted due to the ACA algorithm's ability to adjust the input based on the full dynamics and, for example, allowing the robot to strafe along a wall and only alter the trajectory when a collision is predicted. The BRF is more conservative with a repulsive force always applied with a magnitude based on velocity and distance to the wall. In the current implementation, the forces were tuned to be able to travel through the narrow corridors, which are problematic for potential-field algorithms [23], while still being able to decelerate the robot to a stop when it approaches a wall at high speed. Although it is predicted that the

BRF could allow higher operating speeds in less constrained environments, in applications like search-and-rescue a tightly constrained environment can be expected.

5.8 Conclusions and Future Work

This paper studied UAV-pilot performance with and without the assistance of collision avoidance algorithms. A comparison was done between a feedforward-based collision avoidance algorithm, a basic risk field (potential field-based) algorithm, and full manual control. Human-subject tests were performed where pilots operated a simulated UAV system running the algorithms through three maze-like environments. In the experiments, the number of collisions, the path length, trial time, and average operating speed were recorded. The experimental results showed that the proposed feedforward-based automatic collision avoidance algorithm is capable of significantly improving a pilot's performance compared to manual control and the basic risk field algorithm for the tele-operation of UAVs.

Further studies would include more extensive comparison of the feedforward-based collision avoidance algorithm to other local collision avoidance methods and field studies of the algorithms on various UAV platforms, including fixed-wing configurations.

5.9 References

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

[2] C. Zhang and J. M. Kovas, "The application of small unmanned aerial systems for precision agriculture: a review," *13*, pp. 693–712, 2012.

[3] P. S. Lin, L. Hagen, K. Valavanis, and H. Zhou, "Vision of unmanned aerial vehicle (UAV) based traffic management for incidents and emergencies," in *12th World Congress on Intelligent Transport Systems*, 2005, pp. 1–12.

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

[5] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, "Supporting wilderness search and rescue using a cameraequipped mini UAV," *J. of Field Robotics*, vol. 25, no. 1-2, pp. 89 –110, 2008.

[6] M. Kumar, K. Cohen, and B. Homchaudhuri, "Cooperative control of multiple uninhabited aerial vehicles for monitoring and fighting wildfires," *Journal of Aerospace Computing, Information, and Communication*, vol. 8, no. 1, pp. 1–16, 2011.

[7] S. Waharte and N. Trigoni, "Supporting search and rescue operations with UAVs," in *IEEE Int. Conf. on Emerging Security Technologies*, 2010, pp. 142–147.

[8] J. Israelsen, M. Beall, D. Bareiss, D. Stuart, E. Keeney, and J. van den Berg, "Automatic collision avoidance for manually tele-operated unmanned aerial vehicles," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 6638–6643.

[9] D. Bareiss, J. van den Berg, and K. K. Leang, "Stochastic automatic collision avoidance for tele-operated unmanned aerial vehicles," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015.

[10] T. M. Lam, H. W. Boschloo, M. Mulder, and M. M. van Paassen, "Artificial force field for haptic feedback in uav teleoperation," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 39, no. 6, 2009.

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

[12] 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.

[13] I. Ulrich and J. Borenstein, "Vfh*: local obstacle avoidance with look-ahead verification," in *IEEE Int. Conf. on Robotics and Automation*, 2000.

[14] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: stochastic trajectory optimization for motion planning," in *IEEE Int. Conf. on Robotics and Automation*, 2011.

[15] A. Bry and N. Roy, "Rapidly exploring random belief trees for motion planning under uncertainty," in *IEEE Int. Conf. on Robotics and Automation*, 2011.

[16] S. Patil, J. van den Berg, and R. Alterovitz, "Estimating probability of collision for safe planning under gaussian motion and sensing uncertainty," in *IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 3238–3244.

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

[18] J. Minguez, L. Montano, and O. Khatib, "Reactive collision avoidance for navigation wtih dynamic constraints," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

[19] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *IEEE Int. Conf. on Robotics and Automation: Workshop on Open-Source Software*, 2009.

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

[21] M. Hollander and D. A. Wolfe, *Nonparametric Statistics*. 1973.

[22] J. Neter, M. Kutner, W. Wasserman, and C. Nachtsheim, *Applied Linear Statistical Models*. McGraw-Hill/Irwin, 1996.

[23] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *IEEE Int. Conf. on Robotics and Automation*, 1991.

CHAPTER 6

DISCUSSION AND FUTURE CONSIDERATIONS

The desire to incorporate mobile robots into applications that may be dangerous or undesirable for humans has led to large bodies of research on collision avoidance. Many approaches have been developed for local obstacle knowledge that react to sensor information. Many of the early local collision avoidance methods made assumptions about the robot being modeled as a kinematic system. This assumption allows for many algorithms to determine obstacle avoidance maneuvers purely based upon relative distance or relative velocity between the obstacle and the robot. However, for many robots operating at high speeds or with large inertial effects, their dynamics should not be ignored. Typically, algorithms that assume kinematic models compensate for the known error between the robot's true model and the kinematic model by making the robot maintain some distance from the wall to prevent collisions. Some applications, such as operating a robot indoors with obstacles constantly in close proximity, could perform better if the robot was allowed to get as close as possible to the obstacles without colliding. These robots can maintain smaller distances to obstacles if their dynamics are known and explicitly considered.

In Chapter 2, the Control Obstacle was developed and shown for reciprocal collision avoidance within systems of robots where the robots could have different, nonlinear equations of motion. In this work, the Control Obstacle was shown to be a generalized representation of previous reciprocal collision avoidance methods. An extension to the nonlinear systems was shown and tested in both simulations and experiments. However, the algorithm required careful selection of the time horizon over which trajectories were predicted for the nonlinear systems, such as car-like robots or hovercrafts. If the time horizon was selected to be too small, the robots could create a dead-lock situation where they got too close to each other before attempting to perform evasive actions. Due to their limited control input space, a collision-free input was not always feasible. A time horizon that is too long can also cause the collision-free control input space of the robots to be an empty set. While this

result may not be intuitive at first, consider that the algorithm creates the convex hull of the Control Obstacle with respect to every other robot in the environment. A longer time horizon creates a longer, possibly highly curved, Control Obstacle that sweeps out a larger area as the time horizon increases. Essentially, the algorithm can be overly conservative for a longer time horizon and still lead to a collision by failing to find a feasible change in control input that is collision free. Currently, simple proportional controllers on heading and speed, for example, are implemented to initially test the algorithm. By considering a more complex control system for nonlinear systems that could include sets or sequences of inputs, using Lie algebra for example [1], could allow for more complex trajectories to be considered in the model. An alternative could be to perform reinforcement learning on the system to determine optimal time horizons for varying configurations of robots. Also, given a set of robots and their dynamics, an offline learning algorithm could potentially be implemented to determine the estimated best time horizon for the provided set of robots to avoid collisions [2]–[4].

Chapters 3 and 4 provided an algorithm designed for a single tele-operated robot to avoid collisions. This algorithm incorporates linear halfspace constraints to find a change in input similar to the algorithms in Chapter 2. The algorithm is developed for a stochastic motion and sensing model. However, in practice it can be very difficult to estimate the uncertainty in a motion model that can capture unmodeled dynamics and external disturbances properly. Particularly, with a small unmanned aerial vehicle (UAV) such as a quadcopter, the aerodynamic effects near obstacles can create very large disturbances which are complicated to model [5], [6]. The inclusion of an adaptive model, such as model predictive control [7], [8], could help to update the uncertainty in real time.

In Chapter 4, the sensing uncertainty was developed as a function of the relative distance between an obstacle and the robot. This conveys the fact that many range sensors are more accurate at shorter distances. However, a more generalized approach could be implemented. Through sensing and segmentation of the obstacle into planar faces, each vertex could be assigned a unique covariance value. Through considering the covariance of each vertex, the algorithm could potentially avoid obstacles and respond differently based on the uncertainty of each particular object in the environment. For example, if a set of vertices representing a wall were determined by several sensors' fused data, those vertices would have a lower uncertainty than another wall that was only seen by one obstacle, allowing the robot to maneuver closer to the wall with less uncertainty in its relative position. By computing each vertex's uncertainty with respect to the sensor(s) used to detect it, a generalized sensing

approach could be proposed.

The feedforward model-based collision avoidance made several simplifying assumptions during its derivation. One such assumption was that if the position of the robot at the end of the predicted desired trajectory was collision free, then the robot was collision free along the entire trajectory. This assumption allowed for linear halfspace constraints to define a collision, and a change in input could be calculated through linear programming method. However, there are limitations in the algorithm due to this assumption. Similar to reciprocal collision avoidance methods, the feedforward algorithm in Chapters 3 and 4 requires the selection of a time horizon to look ahead for a collision. This value, if too small, can lead to the robot overshooting its desired position and still potentially leading to a collision (a larger safety buffer through the stochastic algorithm can help alleviate this). However, making the time horizon too long makes the system respond more like an overdamped system and can potentially violate other assumptions in the algorithm's development. Incorporating the velocity into the halfspace constraint through some kind of function, similar to an LQR cost matrix, for example, could provide for a desired position at the time horizon with no velocity component into the obstacle simultaneously.

A second limitation stemming from the position assumption is that nonholonomic robots may not inherently work in this algorithm. For example, imagine a car-like robot driving at an obstacle some extended distance away (see Fig. 6.1(a)). If that obstacle is within a certain distance where the robot has a limited turning radius and is traveling at a high speed, the resultant trajectory from the algorithm will be similar to the previously mentioned case when a controller overshoots its desired position and the endpoint of the position is outside the obstacle. This assumption can also fail to avoid collisions when a quadcopter is provided a yaw input at the same time as a large roll and/or pitch command (see Fig. 6.1(b)). This effect was observed in the first study in Chapter 5. A naive approach to solve this could be to use the maximum penetrating point on the trajectory to define the halfspace constraint, but this could create more halfspace constraints than there are dimensions of the workspace, possibly leading to an infeasible linear programming problem where no collision-free input can be found due to an overconstrained optimization problem.

In Chapter 5, studies were performed to quantify the human operator performance using the automatic collision avoidance (ACA) algorithm in Chapters 3 and 4, manual control, and the basic risk field (BRF) [9]. The ACA algorithm was found to perform better than both manual control and the BRF. In [9], however, the BRF was developed for a velocity-controlled robot with a simplified model. In this dissertation, the studies

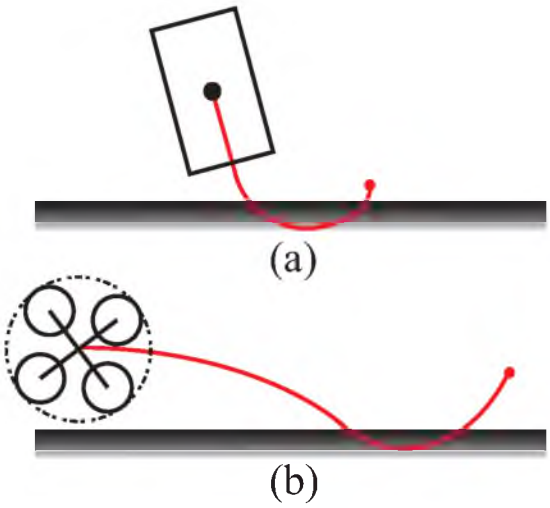

Figure 6.1: Two scenarios in which the algorithm can fail to avoid collisions provided non-holonomic constraints on a (a) car-like robot or a (b) quadcopter that can yaw.

were performed provided acceleration inputs on the simulated quadcopter. Further studies could be beneficial to compare the two algorithms in more depth. First, a study could be performed to determine what gains provide optimal performance for the BRF with acceleration-controlled robots. The ACA algorithm in this dissertation could also have studies performed to determine users' preferred time-horizon for the feedforward prediction. These two user-selected values, the BRF gains and ACA time-horizon, could then be compared again to further verify the results of the studies presented in this dissertation. A set of studies could also be performed using the velocity-controlled robots which the BRF was originally designed for in order to provide a variety of robots for the configuration.

6.1 References

[1] G. Lafferriere and H. Sussmann, "Motion planning for controllable systems without drift.," in *IEEE Int. Conf. on Robotics and Automation*, 1991, pp. 1148–1153.

[2] A. Rai, F. Meier, A. Ijspeert, and S. Schaal, "Learning coupling terms for obstacle avoidance," in *IEEE/RAS Int. Conf. on Humanoid Robots*, 2014, pp. 512–518.

[3] D. Webb, K. Crandall, and J. van den Berg, "Online parameter estimation via real-time replanning of continuous gaussian POMDPs," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 5998–6005.

[4] Q. Wang and C. Phillips, "Cooperative collision avoidance for multi-vehicle systems using reinforcement learning," in *IEEE Int. Conf. on Methods and Models in Automation and Robotics*, 2013, pp. 98–102.

[5] C. G. Hooi, F. D. Lagor, and D. A. Payley, "Flow sensing, estimation and control for rotorcraft in ground effect," in *IEEE Aerospace Conf.*, 2015, pp. 1–8.

[6] D. W. Yeo, N. Sydney, and D. A. Paley, "Onboard flow sensing for rotary-wing uav pitch control in wind," in *AIAA Guidance, Navigation, and Control Conf.*, 2016, pp. 1386–1396.

[7] K. Alexis, C. Papachristoc, R. Siegwart, and A. Tzes, "Robust explicit model predictive flight control of unmanned rotorcrafts: design and experimental evaluation.," in *IEEE European Control Conf.*, 2014, pp. 498–503.

[8] A. T. Hafez, A. J. Marasco, S. N. Givigi, M. Iskandarani, S. Yousefi, and C. A. Rabbath, "Solving multi-UAV dynamic encirclement via model predictive control," *IEEE Trans. on Control Systems Technology*, vol. 23, no. 6, pp. 2251–2265, 2015.

[9] T. M. Lam, H. W. Boschloo, M. Mulder, and M. M. van Paassen, "Artificial force field for haptic feedback in uav teleoperation," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 39, no. 6, 2009.

CHAPTER 7

CONCLUSIONS

This dissertation focused on advancing the state-of-the-art of local collision avoidance methods by explicitly considering robots' dynamics, for single- and multi-robot systems. For the growing number of applications for mobile robots, such as unmanned aerial vehicles (UAVs), the robots' usefulness will increase dramatically if they are able to autonomously avoid each other and obstacles even under human supervision. Thus, automatic collision avoidance technology can improve safety as well as broaden applications of many robotic systems.

First, Objective 1 of this dissertation developed a method of reciprocal collision avoidance for robots with dynamics. Previous methods assumed the robots were all the same type with linear equations of motions. This work, presented in Chapter 2, demonstrated the following:

- a unification of all previous reciprocal collision avoidance approaches under a generalized representation using control obstacles
- real time computation rates for over 100 robots
- collision avoidance for nonhomogeneous systems of robots with nonlinear dynamics in both simulation and real-world experiments

Next, Objective 2 of this dissertation included three tasks. The first was the theoretical development of a stochastic algorithm for collision avoidance of a tele-operated unmanned aerial vehicle (UAV). This task demonstrated the following:

- implementation on a simulated quadcopter in two- and three-dimensional environments
- the approach is capable of avoiding collisions provided uncertainty in the motion model and sensing of obstacles

- the approach performed with a lower *true* probability for collision than the estimated probability for collision

For the second task of Objective 2, the feedforward approach for automatic collision avoidance was implemented on a custom-designed quadcopter UAV. The following was demonstrated during this task:

- the stochastic approach of the previous task is applicable to real-world environments with *completely* on-board sensing and computation
- the quadcopter could maneuver in real-world environments without collisions even as the operator was attempting to cause collisions

For the final task in Objective 2, human-subject studies were performed to quantitatively compare pilots' performance using the feedforward approach as well as the basic risk field (a potential field variant) and full manual control. The algorithm in this dissertation was found to perform better, particularly in terms of average robot speed, which is an *extremely* vital aspect of an application, such as search and rescue, where time is critical. Specifically, the following was found:

- the approach in this dissertation showed an improvement over manual control with a 88% decrease in collisions and a 25% increase in average robot speed
- the approach in this dissertation showed an improvement over the basic risk field with a 36.7% in average robot speed

The results of this dissertation advanced the state-of-the-art of local collision avoidance methods by developing algorithms for both multi- and single-robot systems that can avoid collisions with other robots and obstacles in the workspace while explicitly considering the robots' potentially nonlinear dynamics through a feedforward trajectory estimate. The results contribute to the field of research that is continuing to make mobile robots more useful in beneficial applications such as search and rescue.