Navigating Multiple Simple-Airplanes in 3D Workspace

Jamie Snape, *Student Member, IEEE* and Dinesh Manocha

***Abstract*— We present an algorithm for collision-free navigation of multiple flying robots in three-dimensional workspace. Our approach extends the model of a *simple car* to a *simple-airplane*, which has constraints on speed and steering angle and includes a configuration variable for the altitude. We use a locally *optimal reciprocal collision avoidance* scheme that computes the trajectory without any collisions or oscillations for each airplane independently. In addition, our algorithm explicitly considers the kinematic and dynamic constraints of a simple-airplane and uses the notion of *variable reciprocity* when choosing velocities to ensure that simple-airplanes that are less constrained take more responsibility for avoiding collisions. We test our approach in two simulations and compute collision-free and oscillation-free trajectories that satisfy the kinematic and dynamic constraints of each simple-airplane.**

I. Introduction

Autonomous aircrafts and unmanned aerial vehicles (UAVs) are increasingly used for different applications including mobile surveillance, environmental monitoring, search and rescue, etc. As low-cost hardware and improved sensor technology becomes available, multiple vehicles are frequently used for tracking dynamic targets or providing three-dimensional coverage [1], [2], [3], [4].

In this paper, we consider the problem of collision-free navigation for multiple three-dimensional mobile robots flying in three-dimensional workspace amongst dynamic obstacles. We use a simplified kinematic and dynamic model for each robot based on the two-dimensional *simple car* model [5], [6], [7], albeit without a reverse gear. Rather than fixing the speed [8] or altitude [9], we allow both these variables to vary continuously and refer to the resulting model as a *simple-airplane*.

There is extensive work on collision-free navigation and coordination amongst multiple robots. However, most of the prior methods are limited to two-dimensional robots moving in a plane or do not take into account the kinematic and dynamic constraints on their motion. Moreover, some methods focus on efficient task allocation or pre-compute the entire path of each robot, rather than perform dynamic collision avoidance and navigation.

Main results: We present a novel algorithm that computes collision-free and oscillation-free motion and satisfies kinematic and dynamic constraints. Moreover, we compute the trajectory of each simple-airplane independently and assume no centralized coordination.

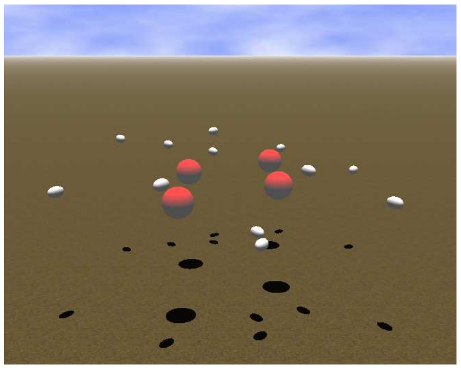


Fig. 1. A screenshot of a simulation of twelve simple-airplanes (smaller white ellipses) navigating through a three-dimensional workspace containing four dynamic obstacles (larger red spheres) which travel at a constant velocity.

Our approach extends the *optimal reciprocal collision avoidance* (ORCA) algorithm [10] to three-dimensional workspaces and uses that to perform local collision avoidance for each simple-airplane. Instead of distributing the load equally amongst the simple-airplanes for collision avoidance, we introduce the notion of *variable reciprocity* between different simple-airplanes. The basic idea is to allocate a higher responsibility to a simple-airplane that has fewer constraints in terms of choosing its velocity. We deal with kinematic and dynamic constraints by sampling the velocities from a reduced set of constraints. The remaining kinematic constraints are satisfied by enumerating a set of pre-computed curves that are defined by the configuration transition equation of the simple-airplane.

We test our approach in two simulations and compute collision-free and oscillation-free trajectories that satisfy the kinematic and dynamic constraints of each simple-airplane, even in the presence of dynamic obstacles.

Organization: The rest of this paper is organized as follows. In Section II, we give a brief overview of related work. We formally outline the model of a simple-airplane and its kinematic and dynamic constraints in Section III. In Section IV, we present the navigation algorithm that uses a three-dimensional collision-avoidance scheme and satisfies the constraints of a simple-airplane. We discuss our implementation and highlight the results in Section V.

II. Previous Work

In this section, we summarize previous work in the areas of motion planning for cars and airplanes with kinematic and

This work was supported in part by ARO contract W911NF-04-1-0088; NSF awards 0636208, 0917040, and 0904990; DARPA/RDECOM contract WR91CRB-08-C-0137; and Intel.

J. Snape and D. Manocha are with the Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA. Email: {snape, dm}@cs.unc.edu. Website (with videos): http://gamma.cs.unc.edu/S-AIRPLANE/.

dynamic constraints as well as relevant work on navigating multiple robots.

A. Planning Under Kinematic and Dynamic Constraints

Some of the earliest work on simplified cars with kinematic constraints appears in Dubins seminal work [11]. The *Dubins car* is restricted to forward motions with a fixed constant speed within a bounded turning radius. The *Reeds-Shepp car* [12] adds a reverse gear to the Dubins car. This was further explored in [13], [14]. A general model with variable speed in any direction subject to a maximum steering angle is presented in [5], [6]. This has been termed a *simple car* by [7] and forms the basis of our simple-airplane model.

The *Dubins airplane* [8] extends the Dubins car into three dimensions with the addition of a configuration variable for altitude. While its speed parallel to the ground remains constant, its rate of change of altitude may vary continuously. Other models for airplanes, for instance [9], choose to fix the altitude and allow the speed of the airplane to vary. It is far less common to allow the altitude as well as the speed to vary, particularly when navigating multiple airplanes or agents amongst each other.

B. Navigating Multiple Robots

Research in multiple airplane coordination and navigation has mainly concentrated on efficient task allocation, for example in [15], or air traffic control management [16], [17], rather than explicitly collision avoidance and local planning. However, there exists a large amount of work on collision-free navigation in a two-dimensional plane in the context of multiple agents or robots.

Many approaches consider other robots as passive moving obstacles [18], [19], [20], [21]. One of the most widely used concepts is that of a *velocity obstacle* (VO) [22], [23], which has also been used in part to navigate a single simulated helicopter [24]. Several variations of the VO formulation have also attempted to incorporate the reactive behavior of other robots [10], [25], [26], [27], [28], [29], [30], some of which take into account either kinematic or dynamic constraints.

Other approaches use follow-the-leader behavior for navigating the robots [31], [32], while there is a large body of research on centrally coordinating the motions of multiple robots, based on centralized methods [7].

III. The Simple-Airplane

In this section, we introduce the simple-airplane based on the simple car [5], [7] and Dubins airplane [8], define its kinematic and dynamic constraints, and introduce the notation used in the rest of the paper.

A. Kinematic Constraints

The simple-airplane, is a four-dimensional system with a configuration $\boldsymbol{q} = (x, y, z, \theta)$ in the configuration space $\mathcal{C} = \mathbb{R}^3 \times \mathbb{S}^1$, where x, y, and z are the coordinates of the origin of the simple-airplane and θ is its orientation, or *yaw*, in the xy-plane relative to the x-axis. It follows that the simple-airplane

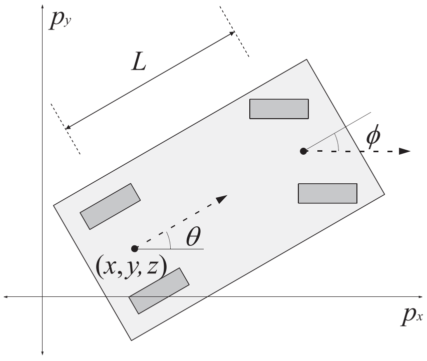


Fig. 2. The simple car, on which the simple-airplane is based. The angle θ is the orientation of the simple car, and the corresponding simple-airplane, in the xy-plane relative to the x-axis, and ϕ is the steering angle. The z-coordinate is fixed for a simple car, but allowed to vary for a simple-airplane. Unlike the simple car, the simple-airplane does not have a reverse gear.

is the simple car in $\mathbb{R}^2 \times \mathbb{S}^1$, illustrated in Fig. 2, augmented with a configuration variable z for altitude. For simplicity, we ignore pitch and roll rotations and other disturbances.

We denote the steering angle of the simple-airplane as ϕ with action variable u_ϕ, its speed (and action variable) parallel to the xy-plane is u_s, and its speed parallel to the z-axis, or *climb rate*, as u_z. For clarity we assume that the distance between the front and rear axles for the equivalent simple car in two dimensions is equal to 1. Therefore, the configuration transition equation $\dot{\boldsymbol{q}}$ of the simple-airplane is given as

$$\begin{aligned} \dot{x} &= u_s \cos\theta, \\ \dot{y} &= u_s \sin\theta, \\ \dot{z} &= u_z, \\ \dot{\theta} &= u_s \tan u_\phi. \end{aligned}$$

Clearly, the system may control the speeds $\dot{x}$ and $\dot{y}$ and yaw rate $\dot{\theta}$ independently from the climb rate $\dot{z}$.

The speed $u_s \in [u_s^{\min}, u_s^{\max}]$ and climb rate $u_z \in [-u_z^{\max}, u_z^{\max}]$ are bounded, as is the steering angle $u_\phi \in [-\phi^{\max}, \phi^{\max}]$ for $\phi^{\max} < \pi/2$. Moreover, a simple-airplane cannot stop or travel in reverse, hence $u_s^{\min} > 0$ (unless, for instance, it represents a blimp or helicopter). We denote this region of allowable actions $\boldsymbol{u} = (u_s, u_z, u_\phi)$ as U. The set V is defined to be the attainable velocities permitted by choosing an action in U.

B. Dynamic Constraints

In addition to its kinematic constraints, the simple-airplane cannot instantaneously increase or decrease its speed or the climb rate arbitrarily fast, nor can it adjust its steering angle discontinuously. Therefore, following the example of [33], we add dynamic constraints $\dot{u}_s \in [-a_s, a_s]$, $\dot{u}_z \in [-a_z, a_z]$, and $\dot{u}_\phi \in [-a_\phi, a_\phi]$. We informally denote this second set of constraints by U' and $\dot{\boldsymbol{u}} = (\dot{u}_s, \dot{u}_z, \dot{u}_\phi) \in U'$. Similarly, V' will be the accelerations that are permitted by U'.

C. Reduced Constraints and Pre-Computed Curves

We use a two-stage approach to compute a velocity that satisfies these constraints, using a reduced set of constraints

$\tilde{U}$ that are easier to calculate, and a set of curves Γ, that may be pre-computed as a preprocess before navigation or simulation commences. In the case of $\tilde{U}$, we consider a window of time τ and define the set of actions at time $t+\tau$ based on the state of the simple-airplane at time t as

$$\tilde{U}(t+\tau) = U \cap \{\boldsymbol{u} \,|\, \boldsymbol{u} = \boldsymbol{u}(t) \oplus \tau \cdot U'\}.$$

The region $\tilde{V}$ to which this corresponds is shown in Fig. 5. While $\tilde{U}$ depends on the time variable t and must be calculated at each time step, the set Γ needs to be computed only once. Each curve $\gamma(\boldsymbol{u})$ in Γ corresponds to the path taken when the simple-airplane performs action $\boldsymbol{u}$ for time τ, and we populate the set by uniformly sampling the range of valid values for each of u_s, u_z, and u_ϕ. For each combination of these three variables, we calculate the values $\dot{x}$, $\dot{y}$, $\dot{z}$, and $\dot{\theta}$ by substituting directly into the configuration transition equation, denoted $\dot{\boldsymbol{q}}$. The change in velocity $\Delta \boldsymbol{v}_\gamma$ of the simple-airplane between the start and the end of the curve γ is pre-calculated and used in the last stage of our algorithm. We pre-calculate about 10^5 curves.

D. Reciprocity Factor

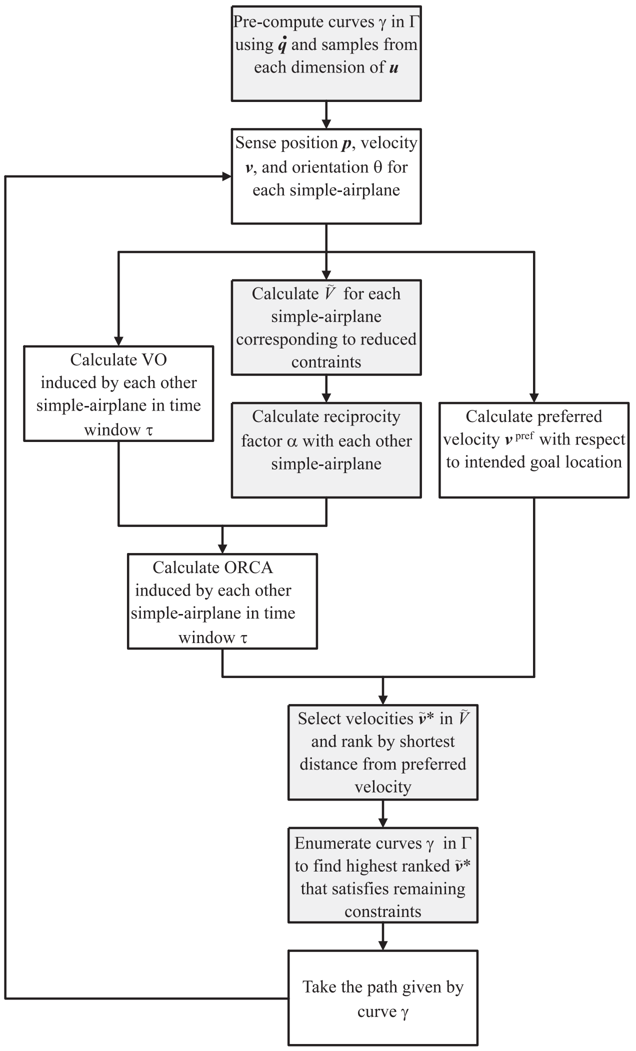
Intuitively, a simple-airplane that has a larger space of attainable velocities $\tilde{V}$, and is therefore less constrained in terms of deviating from its current path, plays a larger role in terms of taking more responsibility for avoiding a collision than a simple-airplane that has a small choice of possible velocities. We refer to this property as *variable reciprocity*, and define the *reciprocity factor* $\alpha \in [0,1]$ as a measure of the amount of responsibility a simple-airplane will take to avoid another simple-airplane. A high value of α denotes that a simple-airplane is less constrained and is able to take a large amount of responsibility for avoiding a collision, a low value represents the opposite. The sum of reciprocity factors α for any two simple-airplanes should always equal 1.

E. Preferred Velocity

During each time step, we use the notion of *preferred velocity* for each simple-airplane $\boldsymbol{v}^{\text{pref}}$. This velocity is either directed towards the final goal position of the simple-airplane or maybe computed using a static roadmap when the environment consists of static obstacles. This velocity may not satisfy the kinematic and dynamic constraints, but is used as an initial guiding velocity in our algorithm for calculating a velocity that indeed satisfies the constraints.

IV. Navigation in 3D Workspace Under Kinematic and Dynamic Constraints

In this section, we outline our approach for navigating multiple simple-airplanes, by extending the concept of *velocity obstacles* (VO) [22] into three dimensions and performing *optimal reciprocal collision avoidance* (ORCA) [10]. We describe how to incorporate an initially reduced, but easily calculated, set of kinematic and dynamic constraints when choosing a velocity using ORCA, and how to deal with the remainder of the constraints by enumerating a set of pre-computed curves that are defined by the same constraints. We also use *variable reciprocity* to ensure that simple-airplanes that are less constrained take more responsibility for avoiding collisions.



Pre-compute curves γ in Γ using $\dot{\boldsymbol{q}}$ and samples from each dimension of $\boldsymbol{u}$
Sense position $\boldsymbol{p}$, velocity $\boldsymbol{v}$, and orientation θ for each simple-airplane
Calculate $\tilde{V}$ for each simple-airplane corresponding to reduced contraints
Calculate VO induced by each other simple-airplane in time window τ
Calculate reciprocity factor α with each other simple-airplane
Calculate preferred velocity $\boldsymbol{v}^{\text{pref}}$ with respect to intended goal location
Calculate ORCA induced by each other simple-airplane in time window τ
Select velocities $\tilde{\boldsymbol{v}}$* in $\tilde{V}$ and rank by shortest distance from preferred velocity
Enumerate curves γ in Γ to find highest ranked $\tilde{\boldsymbol{v}}$* that satisfies remaining constraints
Take the path given by curve γ

Fig. 3. A schematic overview of our approach for the navigation of a simple-airplane amongst other simple-airplanes. Shaded actions indicate steps that ensure that the velocities we calculate satisfy the kinematic and dynamic constraints of a simple-airplane. The first stage consisting of the second, third, and fourth shaded actions calculates and samples the reduced set of constraints $\tilde{U}$, with velocities $\tilde{V}$, and the second stage containing the preprocessing and final shaded actions pre-calculates and enumerates curves in Γ.

A. Overall Approach

The schematic diagram in Fig. 3 outlines our overall approach. In the first stage, the simple-airplane acquires its own position and velocity, and those of surrounding simple-airplanes. It also estimates the kinematic and dynamic constraints of the surrounding simple-airplanes based on their prior motion.

Next, the set of velocities $\tilde{V}$, which correspond to a reduced set of kinematic and dynamic constraints, are calculated for each simple-airplane. Separately, the simple-

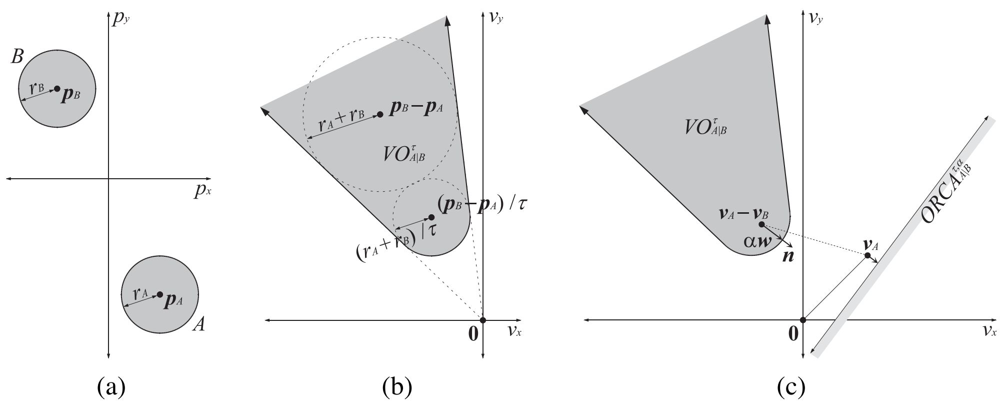


Fig. 4. (a) Two simple-airplanes are bounded by spheres whose centers are $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$ and radii are r_A and r_B, respectively. (b) The shaded area represents the velocity obstacle $VO^{\tau}_{A|B}$ for simple-airplane A induced by B in window of time τ on the xy-plane. The larger circle, with radius $r_A + r_B$, is the Minkowski sum $B \oplus -A$ and the smaller circle, with radius $(r_A + r_B)/\tau$ indicates the amount of truncation due to restricting the size of the window of time to τ. (c) In the xy-plane, the half-plane below the shaded line denoted $ORCA^{\tau,\alpha}_{A|B}$ is the set of permitted velocities specified by optimal reciprocal collision avoidance. The velocity obstacle $VO^{\tau}_{A|B}$ is as defined in (b). The constant α is the *reciprocity factor* which informally represents the amount of load the simple-airplane takes for collision avoidance.

airplane also calculates its preferred velocity $\boldsymbol{v}^{\mathrm{pref}}$ as defined in Subsection III-D. Meanwhile, the VO for the simple-airplane induced by the other simple-airplanes in a window of time τ are constructed.

Using the sets $\tilde{V}$, the reciprocity factor α for the simple-airplane with respect to each other simple-airplane may be calculated, followed by the permitted velocities *ORCA* for the simple-airplane defined by ORCA for a window of time τ. The velocities $\tilde{\boldsymbol{v}}^*$ closest to $\boldsymbol{v}^{\mathrm{pref}}$ that lie within $\tilde{V} \cap ORCA$ are then selected by sampling $\tilde{V}$, and they are ranked by shortest Euclidean distance, in the velocity space, from the preferred velocity. More precisely, if a velocity $\tilde{\boldsymbol{v}}^*_i$ is ranked at position i then $\|\tilde{\boldsymbol{v}}^*_i - \boldsymbol{v}^{\mathrm{pref}}\|_2 \leq \|\tilde{\boldsymbol{v}}^*_j - \boldsymbol{v}^{\mathrm{pref}}\|_2$ for all $j > i$.

Finally, the remaining kinematic constraints that are not used in the calculation of $\tilde{V}$ are satisfied by enumerating the set of pre-computed curves Γ, a subset of all valid paths defined by those constraints. If the first ranked velocity $\tilde{\boldsymbol{v}}^*$ (or one within a small threshold) can be attained by taking a path defined by a curve $\gamma \in \Gamma$, then that velocity is valid and the simple-airplane takes that path. Otherwise it tests the next ranked $\tilde{\boldsymbol{v}}^*$ against the curves Γ and so on, until a match is found.

B. Velocity Obstacles

The concept of a *velocity obstacle* (VO) was originally introduced for navigating a circular robot amongst multiple moving obstacles in two dimensions by [22]. We extend that notion to define a three-dimensional formulation for convex polytopes in the following manner.

Let A be a simple-airplane and let B be an obstacle moving in the space $\mathbb{R}^3$. We deem A and B to have collided if their convex hulls intersect. This is a conservative and sufficient condition, and we use it because it simplifies the derivation. Referring to Fig. 4(a), let $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$ denote the current positions of the reference points of the convex hulls of A and B, respectively, and let $\boldsymbol{v}_B$ be the velocity of moving obstacle B. It follows that the VO for simple-airplane A induced by moving obstacle B for the window of time τ, written $VO^{\tau}_{A|B}$, is the set of all velocities of A that will cause a collision between A and B at some moment before time τ, assuming that B maintains its velocity $\boldsymbol{v}_B$.

More precisely, let $A \oplus B = \{\boldsymbol{a} + \boldsymbol{b} \,|\, \boldsymbol{a} \in CH(A), \boldsymbol{b} \in CH(B)\}$ be the Minkowski sum of the convex hulls of two simple-airplanes A and B, and let $-A = \{-\boldsymbol{a} \,|\, \boldsymbol{a} \in CH(A)\}$ denote the convex hull of A reflected in its reference point. Moreover, let $\lambda^{\tau}(\boldsymbol{p}, \boldsymbol{v}) = \{\boldsymbol{p} + t\boldsymbol{v} \,|\, t \in [0, \tau]\}$ be a ray starting at position $\boldsymbol{p}$ with direction $\boldsymbol{v}$. It follows that

$$VO^{\tau}_{A|B} = \{\boldsymbol{v} \,|\, \lambda^{\tau}(\boldsymbol{p}_A, \boldsymbol{v} - \boldsymbol{v}_B) \cap B \oplus -A \neq \emptyset\}.$$

By definition, if the simple-airplane A selects a velocity inside the region $VO^{\tau}_{A|B}$, shown in Fig. 4(b), then it may potentially collide with moving obstacle B at some future point in time before τ. If the velocity chosen is outside $VO^{\tau}_{A|B}$, then we have a sufficient condition for no such collision within the time interval $[0, \tau]$. The principle of velocity obstacles may therefore be used to navigate a single simple-airplane in the presence of multiple moving obstacles without collisions by having it select a velocity during each time step that is outside the union of all VO induced by the dynamic obstacles [22], [27]. Note that choosing a velocity on the boundary of the VO may result in the simple-airplane and dynamic obstacle grazing each other, which is undesirable for airplanes. Hence, this is explicitly avoided.

Unfortunately, the direct use of VO proves less than satisfactory when navigating multiple simple-airplanes amongst each other. Rather than passive moving obstacles, each of the simple-airplanes is an active decision-making entity and similarly adjusts its velocity with respect to the environment. Hence, for any pair of simple-airplanes, when each chooses a new velocity outside the VO of the other, their previous velocities become valid with respect to the VO of the new velocities. This causes undesirable oscillations that the simple-airplanes may be unable to resolve [28].

C. Optimal Reciprocal Collision Avoidance

To incorporate the reactive behavior of simple-airplanes with respect to each other and avoid oscillations, we use the

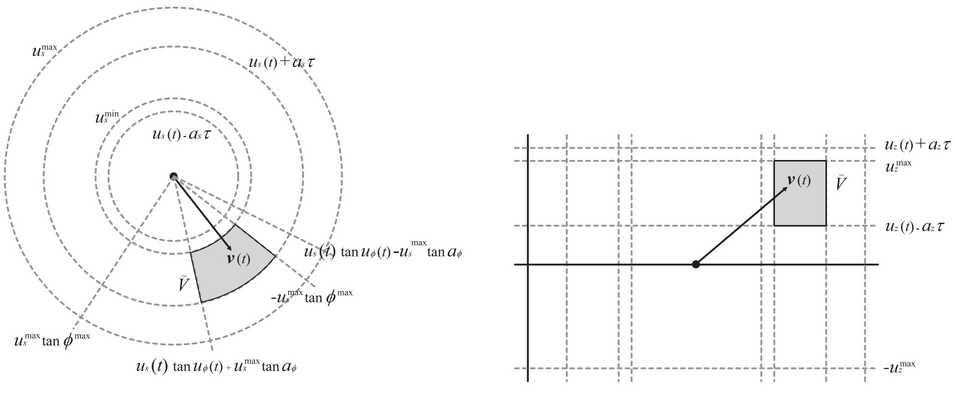


Fig. 5. The set of velocities $\tilde{V}$, corresponding to the reduced set of kinematic and dynamic constraints $\tilde{U}$ from which velocities are sampled, is highlighted by the shaded area. The xy-plane is shown in (a) and the xz-plane is shown in (b). The velocity at time t is $\boldsymbol{v}(t)$ with speed $u_s(t)$, climb rate $u_z(t)$, and steering angle $u_\phi(t)$. The concentric circles in (a) and vertical lines in (b) represent the absolute maximum and minimum speeds, and the maximum and minimum speeds attained by accelerating or decelerating from $u_s(t)$ at the maximum rate for τ time. The horizontal lines in (b) correspond to the analogous values for climb rate, and the rays directed from the center of the circles in (a) are the equivalent values for steering angle.

optimal reciprocal collision avoidance (ORCA) algorithm introduced in [10]. The set of permitted velocities for ORCA are defined geometrically as follows.

Let simple-airplanes A and B have velocities $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$, respectively, and let the velocity obstacle of A induced by B for the window of time τ be $VO^{\tau}_{A|B}$. Moreover, let the relative velocity $\boldsymbol{v}_A - \boldsymbol{v}_B$ be in $VO^{\tau}_{A|B}$, from which it follows that A and B will collide at some point in time before τ if they maintain their current velocities. Let $\boldsymbol{w}$ be the vector from $\boldsymbol{v}_A - \boldsymbol{v}_B$ to the closest point of the boundary of the velocity obstacle $\partial VO^{\tau}_{A|B}$, as indicated in Fig. 4(c), such that

$$\boldsymbol{w} = \big(\operatorname*{arg\,min}_{\boldsymbol{v} \in \partial VO^{\tau}_{A|B}} \|\boldsymbol{v} - (\boldsymbol{v}_A - \boldsymbol{v}_B)\|_2\big) - (\boldsymbol{v}_A - \boldsymbol{v}_B),$$

and let $\boldsymbol{n}$ be the outward normal of $\partial VO^{\tau}_{A|B}$ at the point $(\boldsymbol{v}_A - \boldsymbol{v}_B) + \boldsymbol{w}$. It follows that $\boldsymbol{w}$ is the smallest change to the relative velocity $\boldsymbol{v}_A - \boldsymbol{v}_B$ that will provide a sufficient condition for no collision during the window of time τ. In order to allow the two simple-airplanes to share the responsibility for avoiding a collision, we let A adjust its velocity by $\alpha\boldsymbol{w}$, for the given reciprocity factor $\alpha \in [0, 1]$, as defined in Section III, with the assumption that B takes care of the remaining fraction $1 - \alpha$. It follows that the set of permitted velocities for a simple-airplane A induced by simple-airplane B for the window of time τ, denoted $ORCA^{\tau,\alpha}_{A|B}$ is the half-space in the direction of $\boldsymbol{n}$ starting at the point $\boldsymbol{v}_A + \alpha\boldsymbol{w}$. Formally, the definition is

$$ORCA^{\tau,\alpha}_{A|B} = \{\boldsymbol{v} \,|\, (\boldsymbol{v} - (\boldsymbol{v}_A + \alpha\boldsymbol{w})) \cdot \boldsymbol{n} \geq 0\}.$$

The value of $\alpha = \frac{1}{2}$ used by [10] prescribes that A and B take equal responsibility. It is important to note that while $VO^{\tau}_{A|B}$ contains velocities that will cause a collision in the window of time τ, the velocities within $ORCA^{\tau,\alpha}_{A|B}$ are those which are collision-free for time τ.

D. Kinematic and Dynamic Constraints

The velocities reachable during a window of time τ are restricted by the kinematic and dynamic constraints of the simple-airplane. Recalling the definition in Subsection III-C, we initially consider a reduced set of kinematic and dynamic constraints $\tilde{U}$ with velocities $\tilde{V}$ when choosing from the set of permitted velocities in $ORCA^{\tau,\alpha}_{A|B}$ (see Fig. 5). The attainable velocities that are permitted by optimal reciprocal collision avoidance are therefore $\tilde{V}_A \cap ORCA^{\tau,\alpha}_{A|B}$.

Before taking velocity samples from $\tilde{V}_A \cap ORCA^{\tau,\alpha}_{A|B}$, we first need to choose a value for the reciprocity factor α. This is based on the relative volumes in velocity space of the regions $\tilde{V}_A$ and $\tilde{V}_B$ of the simple-airplanes A and B. The larger the volume of the region for each airplane, the less constrained it is when choosing its next velocity. Hence,

$$\alpha = \operatorname{vol}\tilde{V}_A \,/\, (\operatorname{vol}\tilde{V}_A + \operatorname{vol}\tilde{V}_B).$$

In the unlikely event that both $\operatorname{vol}\tilde{V}_A$ and $\operatorname{vol}\tilde{V}_B$ are zero, then both of the simple-airplanes are entirely constrained. This renders any calculation of $\tilde{V}_A \cap ORCA^{\tau,\alpha}_{A|B}$ meaningless as there is only one velocity that each simple-airplane can take regardless of the value of α. Therefore, this does not affect our definition of the reciprocity factor α. Since α is completely defined by $\tilde{V}_A$ and $\tilde{V}_B$, the notation for the attainable velocities permitted by the optimal reciprocal collision avoidance strategy may be simplified to $\tilde{V}_A \cap ORCA^{\tau}_{A|B}$.

E. Local Planning

After choosing the velocities $\tilde{\boldsymbol{v}}^*$ from $\tilde{U}_A \cap ORCA^{\tau}_{A|B}$ and ranking them with respect to the least Euclidean distance from the preferred velocity $\boldsymbol{v}^{\mathrm{pref}}$, the final task is to satisfy the remaining kinematic constraints not in $\tilde{U}_A$. We use the set of pre-computed curves Γ, defined in Subsection III-C, and enumerate each curve $\gamma \in \Gamma$ comparing the velocity $\boldsymbol{v}(t) + \Delta\boldsymbol{v}_\gamma$ obtained at the end of the curve with our highest ranked velocity $\tilde{\boldsymbol{v}}^*$. If there exists a curve γ that allows $\tilde{\boldsymbol{v}}^*$, or a velocity sufficiently close to $\tilde{\boldsymbol{v}}^*$ in velocity space, to be attained, then the velocity is valid and the simple-airplane chooses that path to make collision-free progress towards its goal. If not, we select the next ranked velocity and repeat the process, continuing until we find a valid combination of velocity $\tilde{\boldsymbol{v}}^*$ and path γ.

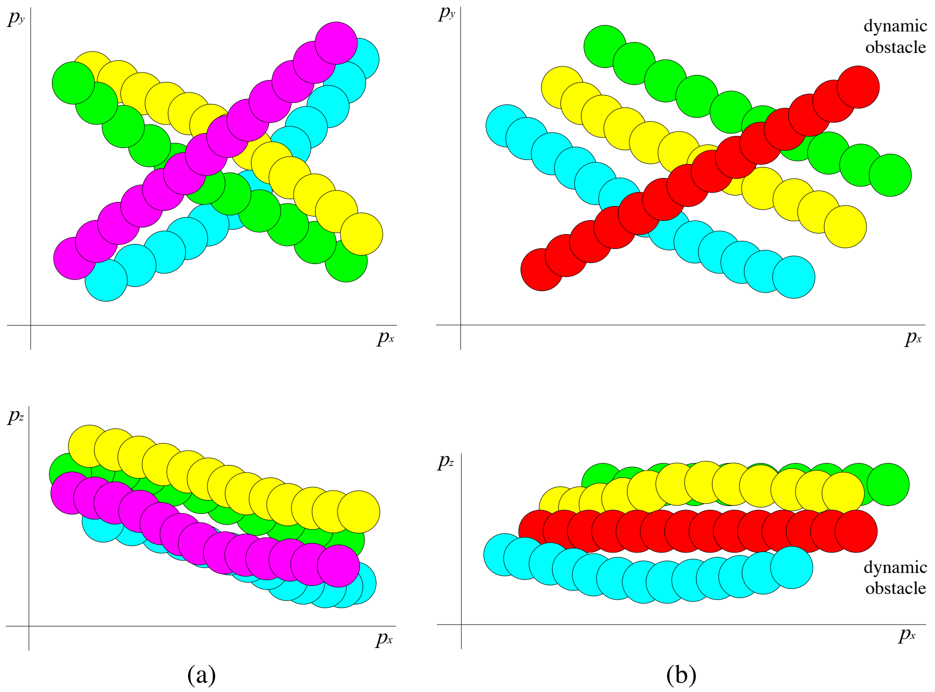


Fig. 6. The traces of the simulated simple-airplanes in the *sphere* scenario (a) and the *dynamic obstacle* scenario (b) for the xy-plane (top) and the xz-plane (bottom). The positions of the simple-airplanes every 10 time steps are shown with a disc. Later positions are drawn on top of earlier positions. The trajectory of each simple-airplane is denoted by a different color, with the red discs in (b) corresponding to the trajectory of the dynamic obstacle.

V. Implementation and Simulation

In this section, we describe our implementation of the approach presented above and report simulated results from several scenarios.

A. Implementation Details

We implemented our approach as a C++ framework on a 2.53 GHz Intel *Core 2 Duo* dual-core processor with 4 GiB of memory running the Mac OS X 10.6 *Snow Leopard* operating system. The calculations for each individual simple-airplane were performed in separate parallel processes using *Grand Central Dispatch*, a programming model and runtime architecture for multi-core computing in *Snow Leopard*.

B. Simulated Results

The scenarios that we simulated were as follows.

- *Sphere:* Simple-airplanes are placed at points on an imaginary sphere. Their goals are to navigate to a point on the opposite side of the sphere. The simple-airplanes will meet and have to negotiate around each other towards the middle of the sphere.
- *Dynamic obstacle:* One or more dynamic obstacles travel at a constant velocity across the environment. The simple-airplanes have to cross the path of the obstacles to reach their goals, while avoiding collisions.

We assume that each simple-airplane has full knowledge of the kinematic and dynamic constraints of the other simple-airplanes, and can easily identify a dynamic obstacle from a cooperating simple-airplane.

Traces of four simple-airplanes for the *sphere* scenario are shown in Fig. 6(a). The paths computed by the simple-airplanes are collision-free and contain no oscillations. They are smooth and direct, with no sudden changes in direction that would indicate that kinematic constraints have been violated. Moreover, the simple-airplanes are not restricted to a fixed plane and change altitude where necessary, clearly using the positions and velocities of the other simple-airplanes to plan their path.

In Fig. 6(b), the *dynamic obstacle* scenario demonstrates that our approach can deal with an entity that is entirely constrained and unable, or unwilling, to adjust its motion due to the proximity of simple-airplanes. Here, the three simple-airplanes detect that the region of attainable velocities of the single dynamic obstacle is a point and of zero volume, therefore naturally obtaining a reciprocity factor of $\alpha = 1$, which corresponds to taking full responsibility for avoiding a collision, without requiring the addition of a special case to our formulation. The dynamic obstacle nominally has a reciprocity factor $\alpha = 0$, indicating that it takes no responsibility for avoiding the simple-airplane.

Videos of expanded versions of the *sphere* scenario with sixteen simple-airplanes and the *dynamic obstacle* scenario with twelve simple-airplanes and four dynamic obstacles, as shown in Fig. 1, are available at http://gamma.cs.unc.edu/S-AIRPLANE/ and via http://www.youtube.com/gammaunc.

VI. Limitations and Conclusions

In this paper, we have introduced the *simple-airplane* and presented an extension of the *optimal reciprocal collision avoidance* (ORCA) algorithm in three dimensions to allow multiple simple-airplanes to navigate amongst each other. In practice, our approach is able to generate collision-free and oscillation-free paths that satisfy the underlying kinematic and dynamic constraints.

Key features of our algorithm are the consideration of most kinematic and dynamic constraints of the simple-airplane when choosing a velocity from within the permitted velocities provided by the ORCA algorithm and the enumeration of pre-computed curves to confirm that the new velocity meets all remaining kinematic constraints. We also account for reciprocity in our formulation to prevent undesirable oscillations, and by incorporating kinematic and dynamic constraints, the idea of *variable reciprocity* is introduced to ensure that simple-airplanes that are less constrained take more responsibility for avoiding collisions. Moreover, the simple-airplanes are restricted to neither constant speed nor fixed altitude as is the case in many approaches.

Our current implementation ignores both roll and pitch of airplanes, but in principle, we can easily extend it to add extra kinematic constraints. We also ignore some external factors such as wind and drag, which may influence the paths chosen by the simple-airplane. Furthermore, we currently assume that each simple-airplane has perfect sensing. We could potentially use the approach of [34] to incorporate uncertainty in position and velocity.

While our formulation is capable of handling moving obstacles, we do not consider static obstacles and the resulting complex environments, containing, for instance, buildings or higher terrain. We would also need to add to our approach a global planner to direct the simple-airplanes to their goals.

Given that each simple-airplane essentially plans its path independently, by only observing other simple-airplanes and dynamic obstacles, there is opportunity to further exploit the parallel nature of this approach by performing some calculations on a GPU, possibly using the *OpenCL* API. Another possibility would be a decentralized approach for the navigation of real flying robots such as a *Blimpduino* blimp [2] or *Mikrokopter* quadrotor helicopter [4].

REFERENCES

[1] P. Cheng, J. Keller, and V. Kumar, "Time-optimal UAV trajectory planning for 3D urban structure coverage," in *Proc. IEEE RSJ Int. Conf. Intell. Robot. Syst.*, 2008, pp. 2750–2757.

[2] C. Anderson, "Meet the Blimpduino," *Make Mag.*, vol. 19, p. 52, 2009.

[3] P. Cheng, V. Kumar, R. Arkin, M. Steinberg, and K. Hedrick, "Cooperative control of multiple heterogeneous unmanned vehicles for coverage and surveillance," *IEEE Robot. Autom. Mag.*, vol. 16, no. 2, p. 12, 2009.

[4] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 2878–2883.

[5] J.-P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot Motion Planning and Control*, ser. Lect. Notes Contr. Inform. Sci., J.-P. Laumond, Ed. London, U.K.: Springer, 1998, vol. 229, pp. 1–53.

[6] J.-C. Latombe, "A fast path planner for a car-like indoor mobile robot," in *Proc. AAAI Nat. Conf. Artif. Intell.*, 1991, pp. 659–665.

[7] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Pr., 2006.

[8] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a Dubins airplane," in *Proc. IEEE Conf. Decis. Contr.*, 2007, pp. 2379–2384.

[9] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proc. Am. Contr. Conf.*, 2002, pp. 1936–1941.

[10] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Proc. Int. Symp. Robot. Res.*, 2009.

[11] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Amer. J. Math.*, vol. 79, pp. 497–516, 1957.

[12] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pac. J. Math.*, vol. 145, no. 2, pp. 367–393, 1990.

[13] H. J. Sussmann and G. Tang, "Shortest paths for the Reeds-Shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control." Rutgers Univ., Tech. Rep. SYNCON 91-10, 1991.

[14] J.-D. Boissonnat, A. Cérézo, and J. Leblond, "Shortest paths of bounded curvature in the plane," *J. Intell. Robot. Syst.*, vol. 11, pp. 5–20, 1994.

[15] A. Richards, J. Bellingham, M. Tillerson, and J. P. How, "Coordination and control of multiple UAVs," in *Proc. AAIA Guid. Navig. Contr. Conf.*, 2002, pp. 1936–1941.

[16] Y.-J. Chiang, J. T. Klosowski, C. Lee, and J. S. B. Mitchell, "Geometric algorithms for conflict detection/resolution in air traffic management," in *Proc. IEEE Conf. Decis. Contr.*, vol. 2, 1997, pp. 1835–1840.

[17] E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk, "Maximum thick paths in static and dynamic environments," *Comput. Geom.*, vol. 43, no. 3, pp. 279–294, 2010.

[18] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, pp. 23–33, 1997.

[19] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot. Res.*, vol. 21, no. 3, pp. 233–255, 2002.

[20] A. Kushleyev and M. Likhachev, "Time-bounded lattice for efficient planning in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 1662–1668.

[21] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *Proc. IEEE RSJ Int. Conf. Intell. Robot. Syst.*, 2005, pp. 2210–2215.

[22] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[23] F. Large, S. Sckhavat, Z. Shiller, and C. Laugier, "Using non-linear velocity obstacles to plan motions in a dynamic environment," in *Proc. IEEE Int. Conf. Contr. Autom. Robot. Vision*, 2002, pp. 734–739.

[24] J. S. Dittrich, F. Adolf, A. Langer, and F. Thielecke, "Mission planning for small UAV systems in unknown environments," in *Proc. AHS Int. Spec. Mtg. Unmanned Rotorcraft Syst.*, 2007.

[25] Y. Abe and M. Yoshiki, "Collision avoidance method for multiple autonomous mobile agents by implicit cooperation," in *Proc. IEEE RSJ Int. Conf. Intell. Robot. Syst.*, 2001, pp. 1207–1212.

[26] B. Kluge and E. Prassler, "Reflective navigation: Individual behaviors and group behaviors," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 4172–4177.

[27] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 1610–1616.

[28] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2008, pp. 1928–1935.

[29] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "ClearPath: Highly parallel collision avoidance for multi-agent simulation," in *Proc. ACM SIGGRAPH Eurographics Symp. Comput. Animat.*, 2009, pp. 177–187.

[30] D. Wilkie, J. van den Berg, and D. Manocha, "Generalized velocity obstacles," in *Proc. IEEE RSJ Int. Conf. Intell. Robot. Syst.*, 2009.

[31] K. C. Ng and M. M. Trivedi, "A neuro-fuzzy controller for mobile robot navigation and multirobot convoying," *IEEE T. Syst. Man Cyb. B*, vol. 28, no. 6, pp. 829–840, 1998.

[32] S. Carpin and L. E. Parker, "Cooperative motion coordination amidst dynamic obstacles," in *Proc. Int. Symp. Distrib. Auton. Robot. Syst.*, 2002, pp. 145–154.

[33] A. Scheuer and C. Laugier, "Planning sub-optimal and continuous-curvature paths for car-like robots," in *Proc. IEEE RSJ Int. Conf. Intell. Robot. Syst.*, 1998, pp. 25–31.

[34] J. Snape, S. J. Guy, J. van den Berg, and D. Manocha, "The stochastic velocity obstacle: Motion planning for multiple mobile robots and virtual agents," Univ. N. Carolina Chapel Hill, Tech. Rep., 2010.