Chapter 1

Consensus-Based Multipath Planning with Collision Avoidance Using Linear Matrix Inequalities

Innocent Okoloko

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.71288

Abstract

Consensus theory has been widely applied to collective motion planning related to coordinated motion. However, when the collective motion is highly irregular and adversarial, the basic consensus theory does not guarantee collision avoidance by default. As collision avoidance is a central problem of path planning, the incorporation of avoidance into the consensus algorithm is a subject of research. This work presents a new method of incorporating collision avoidance into the consensus algorithm, by applying the concept of constrained orientation control, where orientation constraints are represented as a set of linear matrix inequalities (LMI) and solved by semidefinite programming (SDP). The developed algorithm is used to simulate consensus-based multipath planning with collision avoidance for a team of communicating soccer robots.

Keywords: consensus, path planning, avoidance, optimization, LMI

1. Introduction

Path planning has found practical applications in areas such as entertainment (e.g. robot soccer) [1]; self-driving vehicles (e.g. Google's self-driving cars) [2]; intelligent highways [3], and multiple unmanned space systems [4]. Because of the potential applications, the topic of multipath planning has been studied extensively, for example in [5–11].

The simplicity and potential of consensus algorithms to generate *collective behaviors*, such as *flocking*, *platooning*, *rendezvous*, and other *formation* configurations, make it an attractive choice for solving certain problems in multiagent control. However, the basic consensus algorithm collision avoidance mechanism is not developed for *adversarial* situations (i.e., opposite or attacking motion). To extend the power of the algorithm, it is therefore necessary to develop more powerful collision avoidance capabilities.

Next, we consider the basic approaches to collision avoidance in consensus. Some researchers, for example, [12, 13], approached the avoidance problem by introducing potential forces such as attraction and repulsion. However, the potential force algorithms were not developed for adversarial reconfigurations, for example, vehicles moving in opposite directions. Potential functions also have a problem of getting into local minima, coupled with slow speed of convergence. It is observed in [12] that any repulsion based on potential functions alone is not sufficient to guarantee consensus-based collision avoidance. Moreover, the attitude change maneuver presented in [12] was not developed for three-dimensional space (see [14] for a comprehensive literature survey on this topic).

Thus, in this work, we present an approach which we previously developed [5, 9] for incorporating collision avoidance into the consensus framework by applying quadratically constrained attitude control (Q-CAC), via semidefinite programming (SDP), using linear matrix inequalities (LMI). The main benefit of this approach is that it can solve the collision avoidance problem in adversarial situations and any configurations, and the formulation can be applied to two-dimensional as well as three-dimensional spaces. **Table 1** shows the notation frequently used in this chapter.

Notation	Meaning
x^i	Position vector of vehicle number i
$(x^{ij})^{off}$	Offset vector of vehicles i and j
$\mathbf{x}$	Stacked vector of more than one position vector
$\mathbf{x}^{off}$	Stacked vector of more than one offset vector
u^i, $\dot{x}^i$	Control input of vehicle i
$\mathbf{u}$, $\dot{\mathbf{x}}$	Stacked vector of control inputs of more than one vehicle
$\mathbf{L}$	Laplacian matrix
$\mathcal{S}^m$	The set of $m \times m$ positive-definite matrices
$\mathbf{S}$	Bounding sphere or circle of a vehicle or obstacle
ε	Width of safety region
r^*	Radius of $\mathbf{S}$
r	$r^* + \varepsilon$
v^i	Attitude vector of vehicle i
v^i_{obs}	Obstacle vector of vehicle i
v^{ij}_{obs}	Obstacle vector of vehicle i emanating from vehicle j
D^{ij}	Euclidean distance between vehicles i and j
L^{ij}	Line passing through the mid points of vehicles i and j
ρ^{ij}	Perpendicular bisector of L^{ij} separating vehicles i and j
PL^i	Plane passing through the midpoint of vehicle i
l^{ij}	Line of intersection of PL^i and PL^j
d^i_x	Distance from x^i to l^{ij} (for 3D) or p^{ij}(for 2D)

Notation	Meaning
d_v^i	Distance from v^i to l^{ij} (for 3D) or p^{ij}(for 2D)
z^i	A point on the Z axis of PL^i
p^{ij}	Point of intersection of the lines passing through $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$
N^i	Normal vector perpendicular to x^i, v^i, and z^i
$\mathbf{D}$	Attitude control plant matrix, $\mathbf{D} \in \mathcal{S}^m$
$\otimes$	Kronecker multiplication operator
$\mathbf{A}$	State or plant matrix for dynamics of $\mathbf{x}$
$\mathbf{B}$	Input matrix for dynamics of $\mathbf{x}$ for input $\mathbf{u}$
$\mathbf{F}$	Feedback controller matrix
$\mathbf{K}$	Proportional constant
$\mathbf{I}_p$	Identity matrix of size $p \times p$
Γ	$\Gamma = \mathbf{L} \otimes \mathbf{I}_p$
$\mathbf{H}$	A vector or matrix in the Schur inequality
$\mathbf{R}$	A positive-definite matrix in the Schur inequality
$\mathbf{Q}$	A symmetric matrix in the Schur inequality
η	Positive real number for scaling the consensus term
β	Positive real number for scaling the proportional term

Table 1. Frequently used notation in this chapter.

2. Problem statement

The basic *consensus* problem is that of driving the states of a team of communicating agents to a common value by distributed protocols based on their *communication graph*. The agents (or vehicles) $i (i = 1, \cdots, n)$ are represented by vertices of the graph, whereas the edges of the graph represent communication links between them. Let x_i denote the state of a vehicle i and $\mathbf{x}$ is the stacked vector of the states of all vehicles. For systems modeled by first-order dynamics, the following first-order consensus protocol (or its variants) has been proposed, for example in [12, 13]

$$\dot{\mathbf{x}}(t) = -\mathbf{L}\left(\boldsymbol{x}(\boldsymbol{t}) - \boldsymbol{x}^{\boldsymbol{off}}\right). \tag{1}$$

Consensus is said to have been achieved when $\|x^i - x^j\| \to (x^{ij})^{off}$, as $t \to \infty$, $\forall i \neq j$.

The *consensus-based multipath planning with collision avoidance* problem can be stated as follows: Given a set of vehicles i, with initial positions $x^i(t_0)$, desired final positions x_d^i, at time t_f, a set of obstacles with positions $x_{obs}^j (j = 1, \cdots, m)$, and the Laplacian matrix of their communication graph $\mathbf{L}$ find a sequence of collision-free trajectories from t_0 to t_f such that $x^i(t_f) = x_d^i \forall i$. Protocol (Eq. (1)) on its own does not solve the collision avoidance problem in adversarial

situations. A comprehensive presentation of the necessary mathematical tools for this work (including graph theory and consensus theory) can be found in [14].

3. Solutions

In this section, we develop solutions to the problem stated in Section 2.

3.1. Consensus-based arbitrary reconfigurations

It was shown that for the dynamic system.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{2}$$

there exists a stabilizing feedback controller **F**, such that the protocol

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{F}\mathbf{u} \tag{3}$$

drives $\mathbf{x}$ to $\mathbf{x}_f$ [15]. Here, $\mathbf{x} = [x^1, \cdots, x^n]$ is a stacked vector of the initial positions of the vehicles, $\mathbf{u} = -\Gamma(\mathbf{x} - \mathbf{x}^{off})$, $\Gamma = \mathbf{L} \otimes \mathbf{I}_p$, $\mathbf{I}_p$ is the identity matrix of size $p \times p$, and p is the state dimension of the vehicles.

To begin, we first consider the *reference consensus path planning* problem. To this end, the following protocol is proposed for a *leader-follower* communication graph architecture

$$\mathbf{u} = -\Gamma\left(\mathbf{x} - \mathbf{x}^{off}\right) + \mathbf{K}\left(\mathbf{x}^{off} - \mathbf{x}\right). \tag{4}$$

The corresponding protocol for a *leaderless* architecture is

$$\mathbf{u} = -\Gamma\left(\mathbf{x} - \mathbf{x}^{off}\right) + \mathbf{K}(\mathbf{x}_d - \mathbf{x}), \tag{5}$$

where $\boldsymbol{x}_d \neq \boldsymbol{x}^{off}$ is the desired final position and is different from the formation configuration, $\mathbf{K} = \boldsymbol{\epsilon}\mathbf{I}_n$, $(0 < \boldsymbol{\epsilon} \ll 1)$, and n is the dimension of $\mathbf{x}$.

Theorem 1 The time-varying system (Eq. (2)) achieves consensus.

Proof: see [14].

Figure 1 shows a simulation of consensus-based reconfiguration, using the communication graph in **Figure 2**, which is an example of a *leader-follower* graph. Node 1 is the *leader,* and each of the other nodes is connected to their adjacent neighbors. In **Figure 1**, the dots inside small circles indicate initial positions, whereas the dot in the diamond is the initial position of the leader. The stars indicate desired final positions. The larger circles with dashed lines are positions where collisions occurred, and the diameters of the circles indicate the size of intersection of the safety regions of the vehicles. The simulation proves that for arbitrary reconfigurations, the basic consensus algorithm does not guarantee collision avoidance.

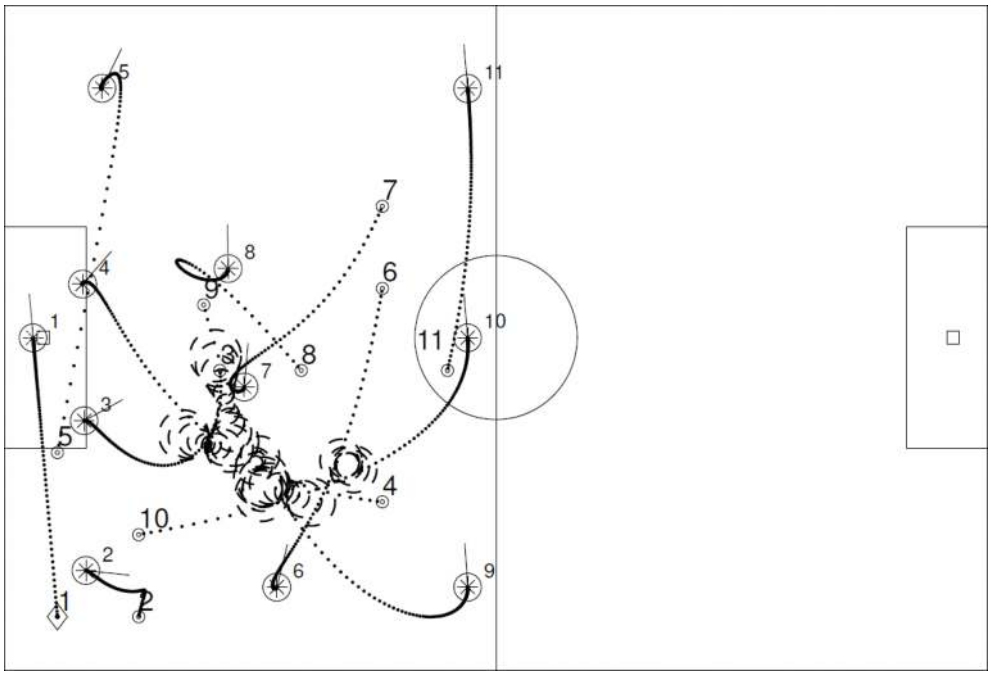

Figure 1. Consensus-based reconfiguration in adversarial situation using topology.

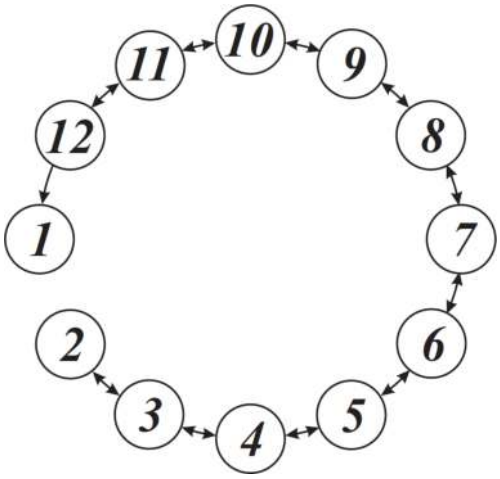

Figure 2. Topology: a leader-follower graph.

3.2. Quadratically constrained attitude control-based collision avoidance

The collision avoidance problem is that of avoiding static obstacles and other moving vehicles while driving the state of a vehicle from one point to another. For simplicity, we approximate a vehicle or an obstacle by **S**, as shown in **Figure 3**. A nonspherical obstacle may be represented by a polygon as shown in **Figure 4**. For the **S**-type obstacle (or vehicle), let the obstacle be centered on a point x_{obs}; it is desired that the time evolution of any vehicle state $x^i(t)$ from t_0 to t_f should avoid the constraint region shown in **Figure 3**.

The feasible region is thus defined by

$$x_{feas} = \{x \in \mathbb{R}^{m\times m} | \|x - x_{obs}\| > r^*\}, \quad m \in \mathbb{R}, \tag{6}$$

where r^* is the radius of **S**, bounded by a safety region of width ε.

There is no direct representation of the nonlinear nonconvex equation (Eq. (6)) as LMI. However, some non-LMI methods, for example, mixed integer linear programming (MILP) [7],

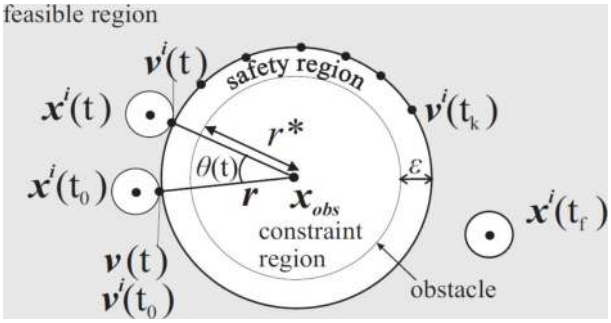

Figure 3. Constrained control problem for a static spherical obstacle.

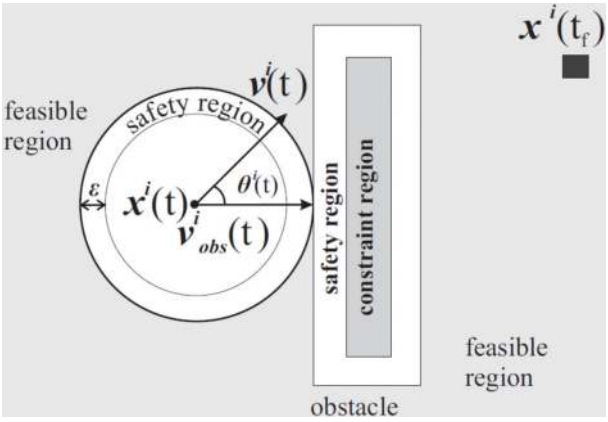

Figure 4. Constrained control problem for static nonspherical obstacle.

have been developed for approximating its solution. In this section, we present an approach, which we previously developed in [5, 9, 10, 14], based on the principles of quadratically constrained attitude control (Q-CAC) algorithm [16], initially developed for the spacecraft attitude control problem.

At any time t, suppose the safety region of vehicle i centered on $x^i(t)$ intersects the safety region of an obstacle, *obs*, centered on x_{obs}. Let $v(t)$ be the unit vector extending from the centre of x_{obs} or $x^i(t)$ in the direction of the point of intersection. The vectors $v(t)$ will be different for each vehicle or obstacle. Considering the case shown in **Figure 3**, assume x_{obs} is known and $v(t)$ is also known in the frame of *obs*. Then, to guide vehicle i safely around the obstacle, define a unit vector $v^i(t)$ in the direction of $v(t)$ in the frame of *obs*. The vector $v^i(t)$ will be regarded as an imaginary vector whose direction can be constrained to change with time. The vector $v^i(t)$ can then be used to find a sequence of trajectories around *obs* which guides i from $x^i(t_0)$ to $x^i(t_f)$ without violating (Eq. (6)).

The problem reduces to the Q-CAC problem. It is desired that the angle θ between $v^i(t)$ and $v(t)$ should be larger than some given angle $\varnothing$, $\forall t$. The constraint is

$$v^i(t)^T v(t) \leq \cos \varnothing, \forall t \in [t_0, t_f]. \tag{7}$$

The idea is to control the angle between the unit vectors $v^i(t)$ and $v(t)$. This implies that one of the vectors $v^i(t)$ or $v(t)$ must remain static, whereas the other moves with time. Vector $v^i(t)$ is used to control the position of the vehicle; therefore, $v^i(t)$ will move with time. The positions of $v^i(t)$ define a trajectory path for $x^i(t)$. Thus, $x^i(t)$ is forced to move on the surface of the safety region bounding $\mathbf{S}$. At some time t_k, $x^i(t)$ will arrive close to a point indicated by $v^i(t_k)$, at which a translation to $x^i(t_f)$ is unconstrained. This is shown by the black dots on the boundary of the safety region in **Figure 4**. To obtain the unit vector $v(t)$, the actual vector extending from the centre of x_{obs} or $x^i(t)$ in the direction of the point of intersection is normalized. After the solution $v^i(t)$ is obtained as a unit vector, $v^i(t)$ is multiplied by $r = r^* + \varepsilon$ to obtain the actual safe trajectory.

Let $\mathbf{v}(t) = [v^i(t)^T v(t)^T]^T$, then the dynamics of $\mathbf{v}(t)$ is defined as

$$\dot{\mathbf{v}}(t) = \mathbf{D}(t)\mathbf{v}(t), \tag{8}$$

where $\mathbf{D} \in \mathcal{S}^{pn}$, p is the dimension of the state vector x^i, and n is the number of vehicles. The above differential equation represents the rotational dynamics of the two vectors contained in $\mathbf{v}(t)$. $\mathbf{D}$ is a semidefinite matrix variable whose contents are unknown. Its purpose is to vary the angle between the two vectors in $\mathbf{v}(t)$ with time while also keeping them normalized.

The discrete time equivalent of the above differential equation is

$$\mathbf{v}(k+1) = \Delta t \mathbf{D}(k)\mathbf{v}(k), \tag{9}$$

where $k = 0, \cdots, N$ ($N\Delta t = t_f$) is the discrete time equivalent of t and Δt is the discretization time-step. To implement Eq. (9), $\mathbf{D}$ is declared in a semidefinite program which chooses the appropriate values to rotate the vectors in $\mathbf{v}(t)$ while satisfying norm constraints. Note in the above discretization of the differential equation, the identity matrix cannot be added to the solution;

instead, the matrix **D** is chosen implicitly to satisfy the rotation. The vectors in $\mathbf{v}(t)$ are unit vectors; they are not translating, but they are rotating and must be preserved as unit vectors.

To enforce the attitude constraint (Eq. (7)) in a SDP, it should be represented as a LMI using the *Schur complement formula* described in [17]. The Schur complement formula states that the inequality

$$\mathbf{H}\mathbf{R}^{-1}\mathbf{H}^{\mathbf{T}} - \mathbf{Q} \leq 0, \tag{10}$$

where $\mathbf{Q}=\mathbf{Q}^{\mathbf{T}}$, $\mathbf{R}=\mathbf{R}^{\mathbf{T}}$, and $\mathbf{R}>0$ are equivalent to and can be represented by the linear matrix inequality

$$\begin{bmatrix} \boldsymbol{Q} & \boldsymbol{H} \\ \boldsymbol{H}^T & \boldsymbol{R} \end{bmatrix} \geq 0. \tag{11}$$

Note that Eq. (7) is equivalent to

$$\underbrace{\begin{bmatrix} v^i(t)^T & v(t)^T \end{bmatrix}}_{v(t)^T} \begin{bmatrix} \boldsymbol{0}_3 & \frac{1}{2}\boldsymbol{I}_3 \\ \frac{1}{2}\boldsymbol{I}_3 & \boldsymbol{0}_3 \end{bmatrix} \begin{bmatrix} v^i(t) \\ v(t) \end{bmatrix} \leq \cos\varnothing, \tag{12}$$

which also implies that

$$\underbrace{\begin{bmatrix} v^i(t)^T v(t)^T \end{bmatrix}}_{v(t)^T} \underbrace{\begin{bmatrix} \boldsymbol{0}_3 & \boldsymbol{I}_3 \\ \boldsymbol{I}_3 & \boldsymbol{0}_3 \end{bmatrix}}_{\mathbf{G}} \begin{bmatrix} v^i(t) \\ v(t) \end{bmatrix} \leq 2\cos\varnothing, \tag{13}$$

Note also that some of the eigenvalues of the **G** in Eq. (13) are nonpositive. To make the matrix positive definite, one only needs to shift the eigenvalues of **G**, by choosing a positive real number μ which is larger than the largest absolute value of the eigenvalues of **G**, then

$$\underbrace{\begin{bmatrix} v^i(t)^T & v(t)^T \end{bmatrix}}_{v(t)^T} \left(\mu\boldsymbol{I}_6 + \begin{bmatrix} \boldsymbol{0}_3 & \boldsymbol{I}_3 \\ \boldsymbol{I}_3 & \boldsymbol{0}_3 \end{bmatrix}\right) \underbrace{\begin{bmatrix} v^i(t) \\ v(t) \end{bmatrix}}_{v(t)} \leq 2\big(\cos\varnothing + \mu\big). \tag{14}$$

Let $\mathbf{M} = \left(\mu\mathbf{I}_6 + \begin{bmatrix} \boldsymbol{0}_3 & \mathbf{I}_3 \\ \mathbf{I}_3 & \boldsymbol{0}_3 \end{bmatrix}\right)^{-1}$, then **M** is positive definite. Therefore, following the Schur complement formula, the LMI equivalent of Eq. (14) is

$$\begin{bmatrix} 2\big(\cos\varnothing + \mu\big) & \boldsymbol{v}(\boldsymbol{t})^{\boldsymbol{T}} \\ \mathbf{v}(\boldsymbol{t}) & \mathbf{M} \end{bmatrix} \geq 0. \tag{15}$$

For collision avoidance, the dynamic system (Eq. (8)) is solved whenever it is required, subject to the attitude constraint (Eq. (15)) and norm constraints $\|v^i(t)\|=1$ and $\|v(t)\|=1$. Thus, the

optimization problem of collision avoidance is essential to find a feasible v^i subject to the following constraints:

$$\mathbf{v}_{k+1} = \Delta t \mathbf{D}(t)\mathbf{v}_k, \tag{16}$$

$$\mathbf{v}_k^T(v_{k+1} - v_k) = 0, \tag{17}$$

$$\begin{bmatrix} 2(\cos\varnothing + \mu) & \mathbf{v}(t)^T \\ \mathbf{v}(t) & \mathbf{M} \end{bmatrix} \geq 0. \tag{18}$$

Eq. (17) is essentially the discrete time version of $\mathbf{v}(t)^T\dot{\mathbf{v}}(t) = 0$ which guarantees that $\mathbf{v}(t)^T\mathbf{v}(t) = 2$, if $\|v^i(0)\| = 1$ and $\|v(0)\| = 1$. This solution works for 2D and 3D spaces. The next step is to extend the formulation to the case of dynamic obstacles. First, consider two vehicles i and j with states $x^i(t)$, $x^j(t)$ and attitude vectors $v^i(t)$, $v^j(t)$, respectively. Collision avoidance requires that they must avoid each other always. As shown in **Figure 5**, any time their safety regions are violated and the point of their intersection in the coordinate frame of i is $v^i_{obs}(t)$.

The avoidance requirements are

$$\theta^i(t) \geq \varnothing \equiv v^i(t)^T v^i_{obs}(t) \leq \cos\varnothing, \tag{19}$$

$$\theta^j(t) \geq \varnothing \equiv v^j(t)^T v^j_{obs}(t) \leq \cos\varnothing, \tag{20}$$

$$\forall t \in [t_0, t_f],$$

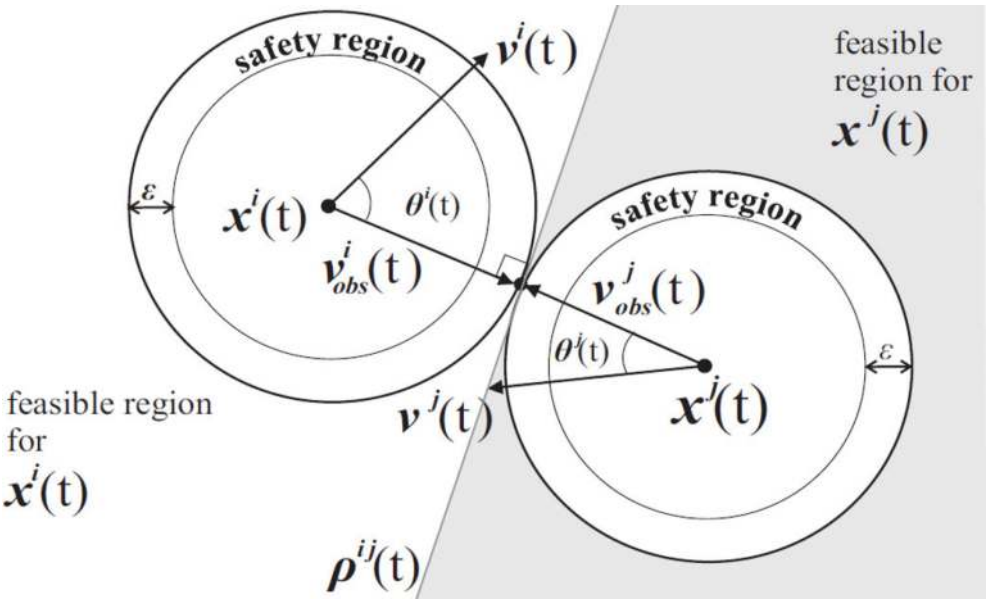

Figure 5. Constrained control problem for dynamic obstacles.

where $\varnothing \geq \pi/2$. For this dynamic situation, it is sufficient to enforce the following avoidance constraints:

$$\begin{bmatrix} 2(\cos\varnothing + \mu) & \begin{bmatrix} v^i(k+2) \\ v^i_{obs}(k+2) \end{bmatrix}^T \\ \begin{bmatrix} v^i(k+2) \\ v^i_{obs}(k+2) \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0, \tag{21}$$

$$\begin{bmatrix} 2(\cos\varnothing + \mu) & \begin{bmatrix} v^j(k+2) \\ v^j_{obs}(k+2) \end{bmatrix}^T \\ \begin{bmatrix} v^j(k+2) \\ v^j_{obs}(k+2) \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0, \tag{22}$$

$$i, j = 1, \cdots, n, i \neq j.$$

Note that $(k+2)$ is used because the optimization is performed two steps ahead of time to ensure that the future trajectories are collision free. However, when this avoidance protocol is applied to dynamic collision avoidance, some vehicle configurations pose challenges and this is considered next.

3.3. Conflict resolution for multiple vehicles

A collision between two vehicles i and j is imminent at time t whenever

$$D^{ij}(t) = \left\| x^i(t) - x^j(t) \right\| \leq \left(r^i + r^j \right), \tag{23}$$

which can be computed using position feedback data determined by onboard or external sensors or communicated among the vehicles.

There are two aspects of collision problems: (i) *collision detection* and (ii) *collision response*. Collision detection is the computational problem of detecting the intersection of two or more objects. This can be done either using sensors or numerically using concepts from linear algebra and computational geometry. Collision response is the initiation of the appropriate avoidance maneuver. In this section, we present methods to detect different configurations of collisions and classify them. Then, an appropriate response technique is developed for each of the collision configurations.

Consider two vehicles i and j, whose current states are $x^i(t)$ and $x^j(t)$ and the desired final states are $x^i(t_f)$ and $x^j(t_f)$. We identify three different basic collision configurations as: (i) *simple collision*; (ii) *head-on collision*; and (iii) *cross-path collision*. Solutions will be developed for each of these configurations, and when combined synergistically, they will provide sufficient collision avoidance behavior for fast collision-free reconfiguration for the team of vehicles.

3.3.1. Detecting and resolving a simple collision

A simple collision problem is any configuration in which $D^{ij}(t) \leq (r^i + r^j)$ and the current vector directions (or attitude vectors) $v^i(t)$ and $v^j(t)$ of vehicles i and j are on different sides of the plane or infinite line $L^{ij}(t)$ passing through the points $x^i(t)$, $x^j(t)$; and the attitude vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ are not parallel. Note that when $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ are not parallel, a point or line of intersection can be computed for both vectors. Examples of simple collision problems are shown in **Figures 5** and **6**.

This is the easiest collision problem to solve because the attitude vectors are already on opposite sides of $L^{ij}(t)$. Considering **Figure 6** (b), the plane or line $\rho^{ij}(t)$ tangent to the point of intersection of both vehicles constrains the current motion spaces of the vehicles to either of the two sides of the plane at time t. A pure optimization-based solution will attempt to search the space on the right side of $\rho^{ij}(t)$ to seek for a point which is closest to the goal of i, and this will be used as the next trajectory. The algorithm will also search the left side of $\rho^{ij}(t)$ to find the next trajectory for j. Once the positions are updated, a new $\rho^{ij}(t)$ is computed.

Indeed, the solution is provided by the basic collision avoidance protocols (Eqs. (21) and (22)) without having to do a set search. It is easy to observe that by expanding the angles $\theta^i(t)$ and $\theta^j(t)$ and choosing the next feasible trajectories $r^*/2$ along the new direction vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$, the new trajectories are bound to satisfy the feasible regions separated by $\rho^{ij}(t)$, provided $\varepsilon^i > r^{*i}$ for any i. The rest of the avoidance strategies developed in the remaining part of this section are attempts to reduce more complex collision configurations to a simple collision configuration.

3.3.2. Detecting and resolving a head-on collision

A head-on collision problem is any configuration in which $D^{ij}(t) \leq (r^i + r^j)$ and $v^i(t)^T v^j(t) \approx \pi$ rad. **Figure 7(a)** illustrates the head-on collision problem.

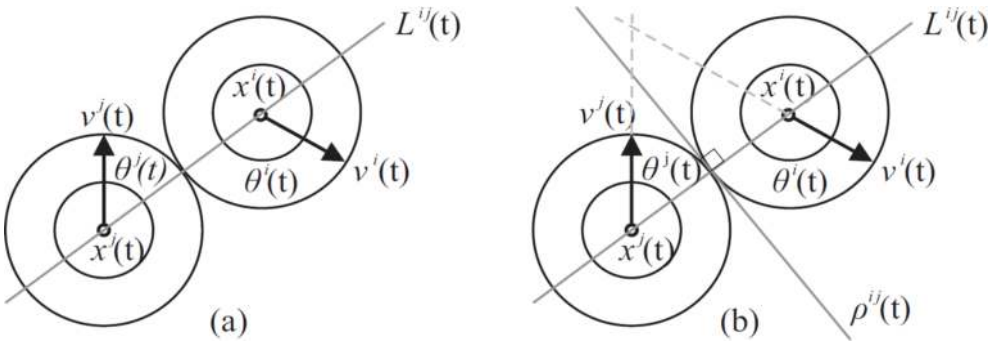

Figure 6. Simple collision problem.

The paths from the current positions $x^i(t)$ and $x^j(t)$ to the goal positions $x^i(t_f)$ and $x^j(t_f)$ lead to a configuration in which the attitude vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ are parallel (or close to parallel) and in opposite directions, in the sense that a point of intersection cannot be computed. **Figure 7(b)–(d)** shows several examples of head-on collision. **Figure 7(b)** is a *direct head-on collision* because the vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ are lying directly on $L^{ij}(t)$. **Figure 7(c)** is an *approximate head-on collision* and **Figure 7(d)** is a head-on collision that can be easily converted to a simple collision configuration.

For the configurations in **Figure 7(b)** and **(c)**, the Q-CAC formulation presented earlier easily solves this problem without any modifications to the algorithm. However, whenever $v^i(t)^T v^i_{obs}(t) \approx 0$ for any i, the optimization algorithm takes some significant time to solve. Even though the resulting trajectory is desirable, this delay is undesirable for real-time collision avoidance. Therefore, whenever this configuration is encountered for any two vehicles, a one-step elementary evasive maneuver is initiated, in which either $v^i(t)$ or $v^j(t)$ is rotated by a small angle $\psi > 0$. This rotation effectively transforms the head-on collision

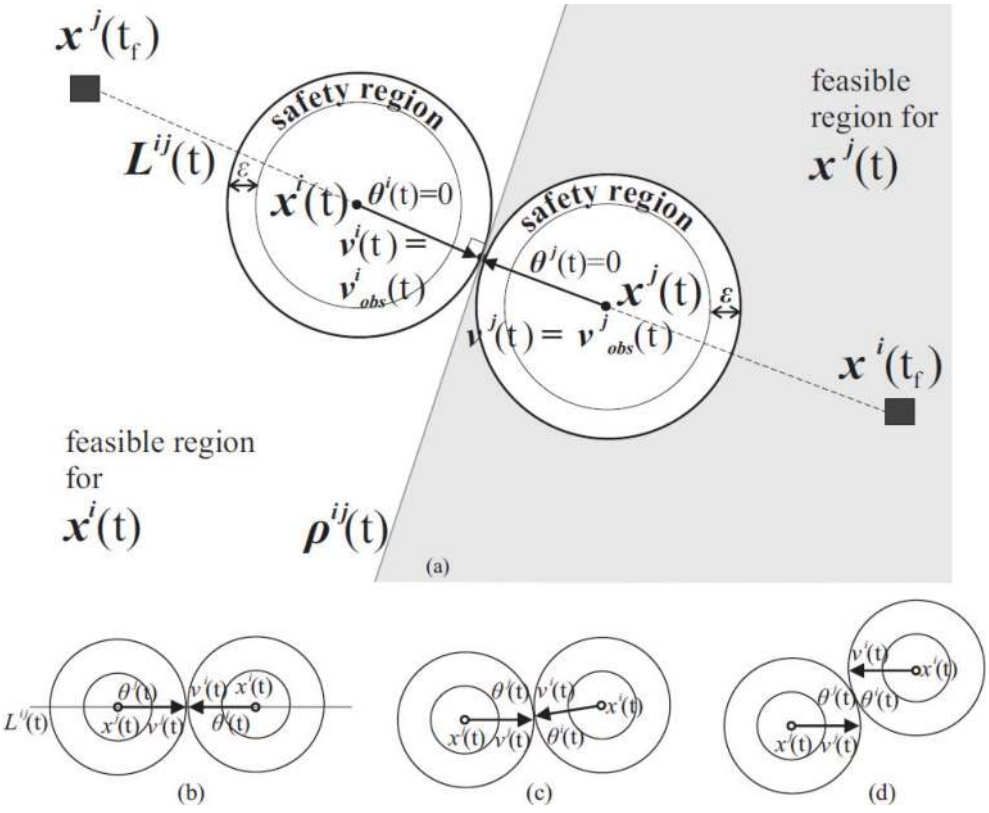

Figure 7. Head-on collision problem.

configuration to a simple collision configuration. Once this is done, the avoidance constraints defined in Eqs. (21) and (22) solve in real time. The trajectory obtained using this strategy for two-vehicle reconfiguration with head-on collision avoidance is shown in [14].

3.3.3. Detecting and resolving cross-path collision for two vehicles

A cross-path collision problem is any configuration in which $D^{ij}(t) \leq (r^i + r^j)$ and the current vector directions $v^i(t)$ and $v^j(t)$ are on the same side of $L^{ij}(t)$, and the attitude vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ are not parallel. Because the vectors are not parallel, a point (for 2D) or line (for 3D) of intersection can be computed for both vectors. **Figure 8** is an example of a cross-path collision problem.

Note that for the avoidance process, the attitude control algorithm attempts to expand the angles $\theta^i(t)$ and $\theta^j(t)$ to an angle = $\pi/2$. Based on this initial configuration, therefore, $v^i(t)$ and $v^j(t)$ will remain parallel or close to parallel, but not in opposite directions. If this continues, the desired goal positions may never be reached, or may be reached after a great deal of effort. To resolve this problem, it is required to determine whether the two vehicles are indeed in a cross-path configuration. The task is therefore to see if there exists a point or line of intersection between $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$, and if such an intersection lies on one side of L^{ij}.

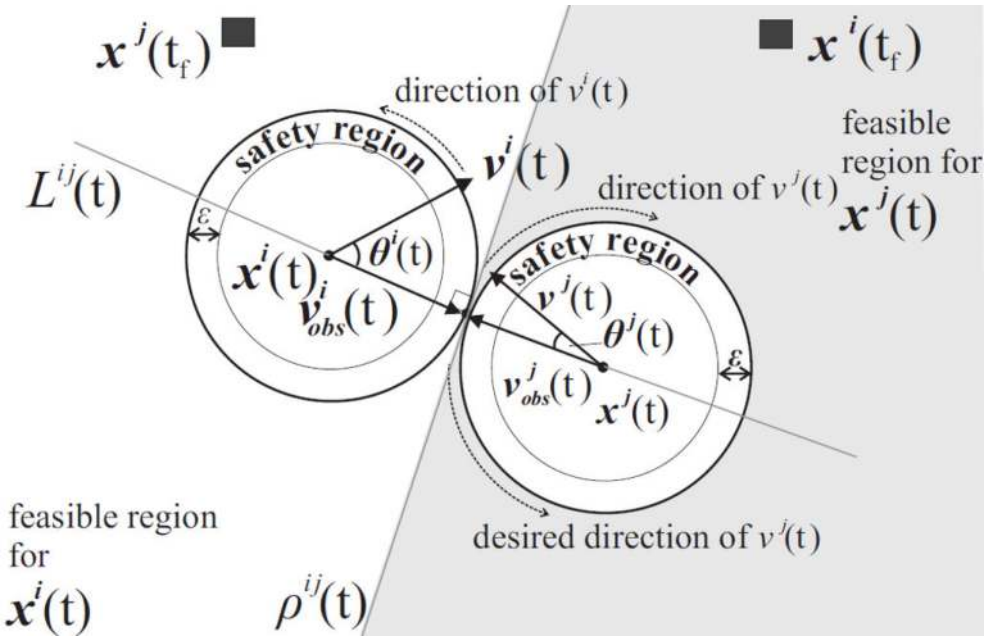

Figure 8. Cross-path collision trajectory.

3.3.4. Determining cross-path collision in 3D and 2D.

To determine cross-path collision between i and j in 3D, two planes PL^i and PL^j are defined, both parallel to the z axes of the world coordinate frame (**Figure 9**). Each plane must contain the vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ as shown in the figure. Therefore, the plane PL^i is defined as the set $(N^i(t), x^i(t), v^i(t), z(t))$, where $z^i(t)$ is a point chosen above or below $x^i(t)$ or $v^i(t)$ on the z axis and $N^i(t)$ is the normal vector perpendicular to $x^i(t)$, $v^i(t)$, and $z^i(t)$. Once $N^j(t)$ is similarly defined, the intersection of the two planes can be computed using techniques from computational geometry. If the two planes are not parallel, the computation of planes' intersection will return a line l^{ij}. Once this line is determined, the next step is check if it is on one side of the plane parallel to the z axis and containing the points $x^i(t)$ and $x^j(t)$.

An easy way to do this is to compute the perpendicular distances from the points $x^i(t)$, $v^i(t)$, $x^j(t)$, and $v^j(t)$, to l^{ij}.

Let the corresponding distances be:

$$d_x^i(t) = \left\| x^i(t) - l^{ij} \right\|, \tag{24}$$

$$d_v^i(t) = \left\| v^i(t) - l^{ij} \right\|, \tag{25}$$

$$d_x^j(t) = \left\| x^j(t) - l^{ij} \right\|, \tag{26}$$

$$d_v^j(t) = \left\| v^j(t) - l^{ij} \right\|. \tag{27}$$

If $d_v^i(t) \le d_x^i(t)$ and $d_v^j(t) \le d_x^j(t)$, then the line of intersection is in front of both vehicles, and a cross-path collision is imminent as shown in **Figure 9(a)** and **(b)**. Otherwise, there is no cross-path conflict as shown in **Figure 9(c)**.

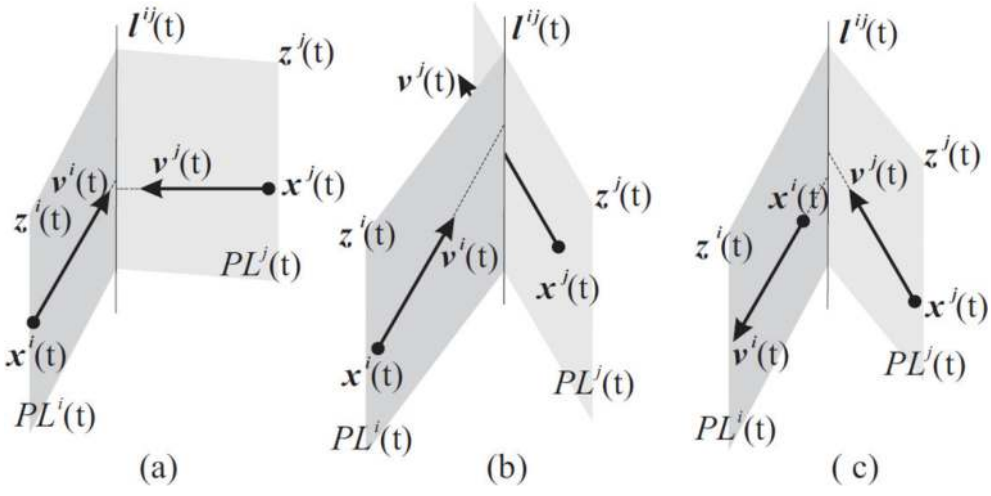

Figure 9. Determination of cross-path collision in 3D.

The analysis is simpler in the 2D case. Instead of l^{ij}, we search for a point p^{ij}, which is the point of intersection of the lines passing through $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ as shown in **Figure 10**. If indeed such a point is found, we use p^{ij}instead of l^{ij}in the previous set of equations.

Figure 11 shows an illustration of the computation of d_x^i and d_v^i for any i.

Figure 12(a) is a cross-path collision configuration, but (b) is a simple collision configuration.

The solution strategy adopted is to convert any cross-path configuration such as **Figure 12** (a) to a simple configuration such as (b). To do this, one only must move either $v^i(t)$ or $v^j(t)$ to the other side of $L^{ij}(t)$ (or onto the line $L^{ij}(t)$). A simple strategy to decide which $v(t)$ should be

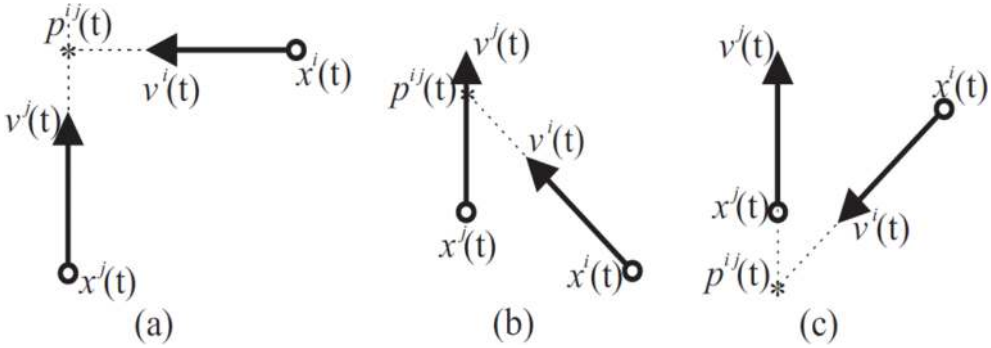

Figure 10. The point p^{ij} is the point of intersection of the infinite lines passing through direction vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$. The position of p^{ij} in relation to both direction vectors determines if a cross-path collision is imminent. If p^{ij} is in front of both vectors as in (a) and (b), then a cross-path collision is imminent; otherwise, no cross-path collision is imminent as in (c).

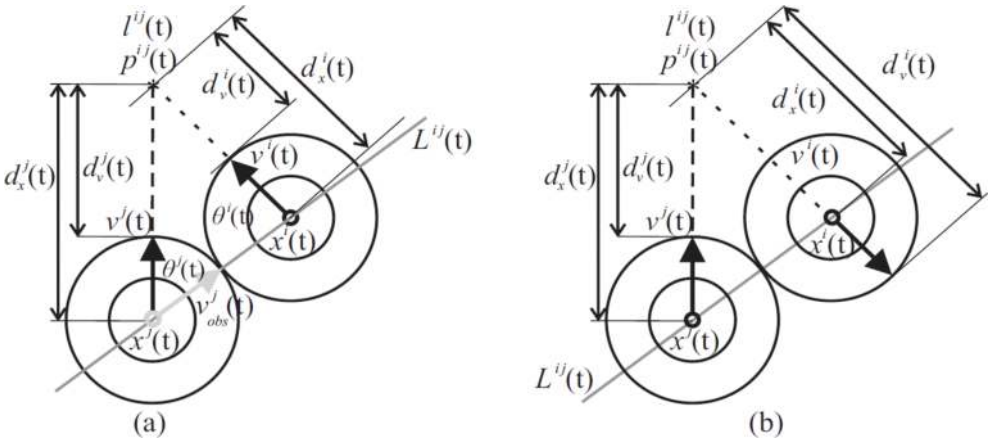

Figure 11. Continuing the explanation from **Figure 10**, d_x^i is the distance from any x^i (vehicle i) top^{ij}, whereas d_v^i is the distance from v^i to p^{ij}, that is, the distance of the outer boundary (where v^i lies) of the safety region of vehicle i to p^{ij}.

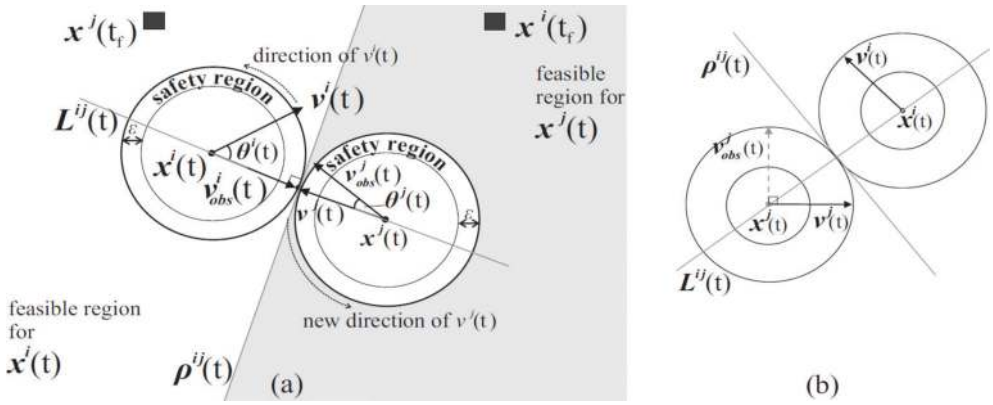

Figure 12. The effects of cross-path conflict resolution.

moved to obtain smoother phase transition is to measure $\theta^i(t)$ and $\theta^j(t)$. If $\theta^j(t) < \theta^i(t)$, then $v^j(t)$ should be moved. This is done by swapping $v^j(t)$ and $v^j_{obs}(t)$, which immediately results in a simple collision reconfiguration. Thereafter, when the Q-CAC algorithm expands $\theta^j(t)$, it is the former $v^j_{obs}(t)$ (which is now the new $v^j(t)$) that moves, whereas the former $v^j(t)$ (which is now the new $v^j_{obs}(t)$) remains static.

Therefore, if a cross-path trajectory is determined, to resolve the problem it is sufficient to swap the variables in one of the avoidance constraints (Eq. (21) or Eq. (22)). For example, Eq. (21) may be left as it is and Eq. (22) is rewritten in the form

$$\begin{bmatrix} 2(\cos\varnothing + \mu) & \begin{bmatrix} v^j_{obs}(k+2) \\ v^j(k+2) \end{bmatrix}^T \\ \begin{bmatrix} v^j_{obs}(k+2) \\ v^j(k+2) \end{bmatrix} & M \end{bmatrix} \geq 0. \tag{28}$$

The trajectories obtained by applying this strategy to cross-path collision avoidance for two vehicles in 2D and 3D are shown in [14].

3.3.5. Resolving cross-path collision for more than two vehicles

If more than two vehicles are involved as shown in **Figure 13**, for any vehicle *i*, whose attitude vector $v^i(t)$ is in a cross-path configuration with vehicles *j* and *k*, we are concerned only about the two bounding obstacle vectors $v^{ij}_{obs}(t)$ and $v^{ik}_{obs}(t)$.

In order not to get into a stalemate situation (undesirable for aircraft), only positive nonzero velocities are required to be generated. We adopt a counterclockwise avoidance measure to achieve this, where, for each vehicle, the left bounding obstacle vector is always chosen as the

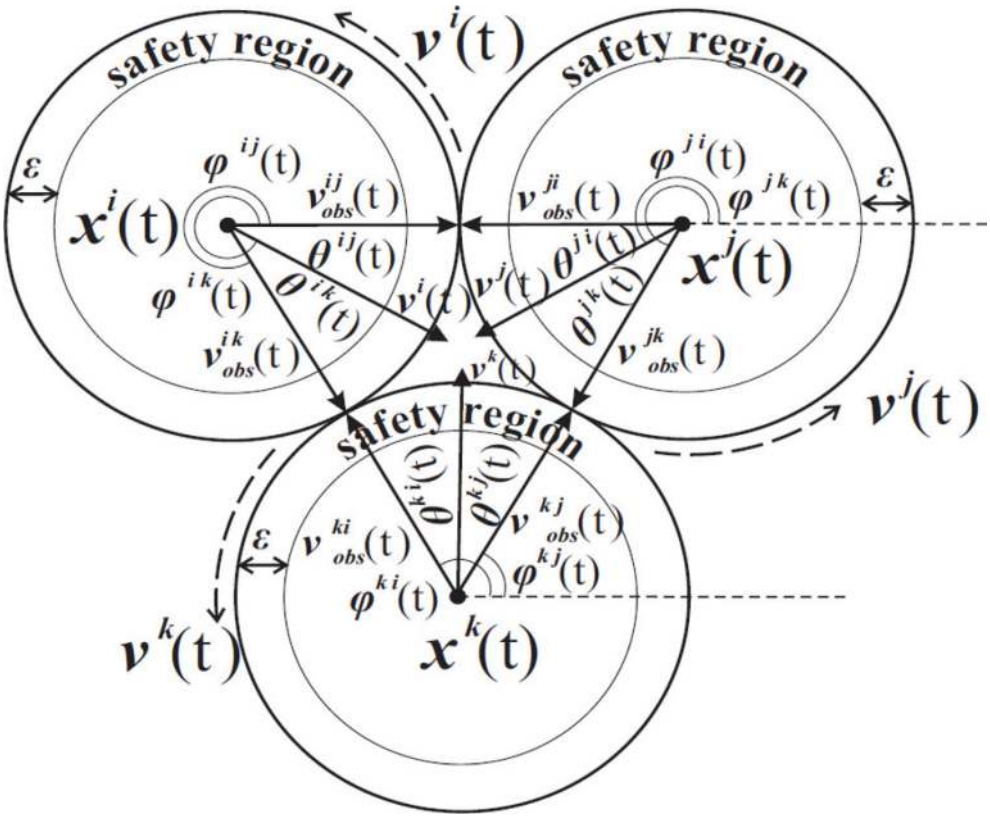

Figure 13. Three-vehicle cross-path trajectory problem.

cross-path obstacle vector for avoidance. For example, for k to turn counterclockwise, it chooses the vector $v^{ki}_{obs}(t)$ to avoid instead of $v^{kj}_{obs}(t)$. Thus, for the configuration of **Figure 13**, the following set of attitude constraints is enforced:

$$\begin{bmatrix} 2(\cos\varnothing+\mu) & \begin{bmatrix} v^{ij}_{obs}(k+2) \\ v^{i}(k+2) \end{bmatrix}^T \\ \begin{bmatrix} v^{ij}_{obs}(k+2) \\ v^{i}(k+2) \end{bmatrix} & M \end{bmatrix} \geq 0, \tag{29}$$

$$\begin{bmatrix} 2(\cos\varnothing+\mu) & \begin{bmatrix} v^{jk}_{obs}(k+2) \\ v^{j}(k+2) \end{bmatrix}^T \\ \begin{bmatrix} v^{jk}_{obs}(k+2) \\ v^{j}(k+2) \end{bmatrix} & M \end{bmatrix} \geq 0, \tag{30}$$

$$\begin{bmatrix} 2\left(\cos\varnothing+\mu\right) & \begin{bmatrix} \boldsymbol{v}_{\boldsymbol{obs}}^{\boldsymbol{ki}}(\boldsymbol{k}+\boldsymbol{2}) \\ \boldsymbol{v}^{\boldsymbol{k}}(\boldsymbol{k}+\boldsymbol{2}) \end{bmatrix}^{T} \\ \begin{bmatrix} \boldsymbol{v}_{\boldsymbol{obs}}^{\boldsymbol{ki}}(\boldsymbol{k}+\boldsymbol{2}) \\ \boldsymbol{v}^{\boldsymbol{k}}(\boldsymbol{k}+\boldsymbol{2}) \end{bmatrix} & \boldsymbol{M} \end{bmatrix} \geq 0. \tag{31}$$

3.4. Consensus with Q-CAC–based avoidance

Once a safe attitude vector $v^i(k)$ is computed at time k for any i, the next position $x^i(k+1)$ is computed as a point a distance $r^{*i}/2$ from the current position, along the vector $v^i(k)$. Note that $v^i(k)$ is normalized to keep the computed control bounded. Whether there are intersections of the safety regions or not, one can guarantee the safety of the algorithm by bounding the control size within the interval $0 < u^i \leq r^{*i}/2$. This means that a vehicle never steps beyond its safety region at any single time step.

Another important consideration is the size of control computed at each time using Laplacian matrices, which is directly proportional to the algebraic connectivity of the communication graph, and inversely proportional to the magnitude of the current time k. This means that, while the early values of **u** are large and therefore unsafe for collision avoidance (and must be bounded), the latter values of **u** are very small and therefore slow down the rate of convergence. One can observe that collisions are less likely to occur in the latter times when the vehicles are closer to their goal positions; consequently, convergence is slower at that time. Therefore, there is need to obtain constantly bounded control **u** which can guarantee both collision avoidance and a high speed of convergence. The following modifications to Eq. (4) and Eq. (5) were proposed in our previous works [5, 9, 14]. For the leader-follower architecture,

$$\mathbf{u} = -\eta \log_{10}(k+1)\frac{\Delta t}{2\lambda_2(\boldsymbol{L})}\Gamma\left(\boldsymbol{x}-\boldsymbol{x}^{\boldsymbol{off}}\right) - \beta \log_{10}(k+1)\frac{\Delta t}{2\lambda_2(\boldsymbol{L})}\mathbf{K}\left(\boldsymbol{x}-\boldsymbol{x}^{\boldsymbol{off}}\right). \tag{32}$$

And for the leaderless architecture,

$$\mathbf{u} = -\eta \log_{10}(\mathrm{k}+1)\frac{\Delta t}{2\lambda_2(\mathbf{L})}\Gamma\left(\boldsymbol{x}-\boldsymbol{x}^{\boldsymbol{off}}\right) - \beta \log_{10}(\mathrm{k}+1)\frac{\Delta t}{2\lambda_2(\mathbf{L})}\mathbf{K}(\boldsymbol{x}-\boldsymbol{x}_{\boldsymbol{d}}), \tag{33}$$

where $\lambda_2(\boldsymbol{L})$ is the second smallest eigenvalue of the Laplacian **L**. The parameter η is for scaling the consensus term and β is for scaling the proportional term in Eqs. (32) and (33). The logarithmic term $\log_{10}(k+1)$ and the term $\frac{\Delta t}{2\lambda_2(\boldsymbol{L})}$ are used to reduce $\|\boldsymbol{u}\|$ when k is small and increase $\|\boldsymbol{u}\|$ when k is large. The choices of parameters η and β should depend on the radius of **S** and safety region ε for each vehicle. Alternatively, one may choose to compute an unbounded **u** using Eqs. (4) or (5), then for each $u^i > \frac{r^{*i}}{2}$, normalize u^i and set $u^i = \frac{r^{*i}}{2}u^i$.

The step-by-step procedure for implementing the algorithm including a flowchart can be found in Ref. [14].

4. Simulation results

To demonstrate the solutions developed in this chapter, we revisit the experiment presented in **Figure 1**. The robots are homogeneous, and **S** for each robot is 85 mm, $\varepsilon = 90$ mm, whereas the dimensions of the soccer pitch are 6050 mm x 4050 mm. In **Figure 14** (a), Eq. (5) was applied with the cyclic communication topology with one leader (**Figure 2**). In **Figure 14** (b), Eq. (33) was applied with a full communication topology (i.e., every vehicle can communicate with each other). The simulation was done with MATLAB R2009a on an Intel® Core(TM)2 Duo P8600 @ 2.40 GHz with 2 GB RAM, running Windows 7. For **Figure 14(a)**, the multipath planning problem took 244 time-steps to solve, resulting in a total computation time of 7.343 s, in which 203 avoidance attempts were made, and there were no collisions. For **Figure 14(b)**, using a full communication topology, the computational time was 0.0131 s, and there were no collisions.

In [14], more simulations and analyses are presented, together with the limitations of this approach, which remains to be explored for future development.

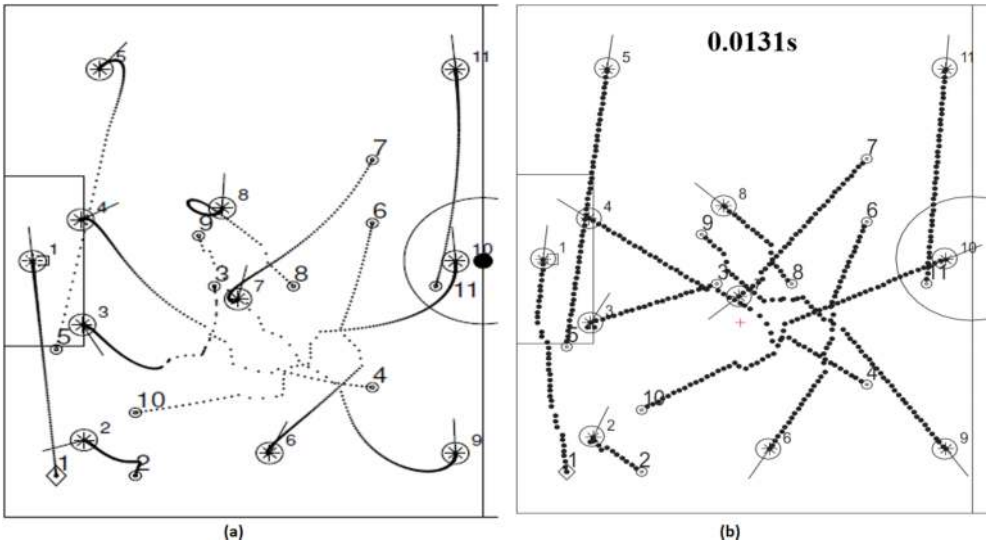

Figure 14. Collision-free reconfiguration: (a) using topology with Eq. (5) and (b) using fully connected graph with Eq. (33).

5. Conclusion

In this chapter, we considered consensus-based multipath planning. An approach to incorporating collision avoidance in adversarial situations in the consensus algorithm by applying Q-CAC is presented. Simulation results are presented here to show that for a sizable number of

vehicles, collision avoidance and fast convergence are guaranteed. Future work will include implementation on a team of mobile robots and autonomous aerial vehicles.

Author details

Innocent Okoloko

Address all correspondence to: okoloko@ieee.org

Department of Electrical Engineering, Universidad de Ingenieria y Tecnologia, Lima, Peru

References

[1] Zickler S, Laue T, Birbach O, Wongphati M, Veloso M. SSL-vision: The shared vision system for the RoboCup small size league. In: RoboCup 2009: Robot Soccer World Cup XIII. Baltes J, Lagoudakis MG, Naruse T, Shiry S, editors. Lecture Notes in Artificial Intelligence. Springer; 2010. p. 425-436. ISBN 978-3-642-11876-0

[2] Waymo. Google's Self-Driving Cars [Internet]. 2016. Available from: https://waymo.com/ [Accessed: August, 2017]

[3] PATH. California Partners for Advanced Transit and Highways [Internet]. 2006. Available from: http://www.path.berkeley.edu/ [Accessed: December, 2009]

[4] Blackwood G, Lay O, Deininger B, Gudim M, Ahmed A, Duren R, Noeckerb C, Barden B. The StarLight mission: A formation-flying stellar interferometer. In: SPIE 4852, Interferometry in Space; 22 August; Waikoloa, Hawaii. SPIE Digital Library; 2002. DOI: http://dx.doi.org/10.1117/12.460942

[5] Okoloko I, Basson A. Consensus with collision avoidance: An LMI approach. In: 5th IEEE International Conference on Automation, Robotics and Applications; 06–08 December; Wellington, NZ. IEEE; 2011. ISBN:9781457703287

[6] Chandler PR, Pachter M, Rasmussen S. UAV cooperative control. In: IEEE ACC; 25–27 June; Arlington, VA. IEEExplore; 2001. p. 50-55. DOI: 10.1109/ACC.2001.945512

[7] Richards A, Schouwenaars T, How JP, Feron E. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. AIAA Journal of Guidance Control and Dynamics. 2002;**25**:755-764. DOI: https://doi.org/10.2514/2.4943

[8] Okoloko I. Consensus based distributed motion planning on a sphere. In: IEEE ACC; 17–19 June; Washington DC. IEEEXplore; 2013. p. 6132-6137. DOI: 10.1109/ACC.2013.6580799

[9] Okoloko I. Path planning for multiple spacecraft using consensus with LMI avoidance constraints. In: IEEE Aerospace Conference; 3–10 March; Big Sky, Montana. IEEEXplore; 2012. p. 1-8. DOI:10.1109/AERO.2012.6187118

[10] Okoloko I, Kim Y. Distributed constrained attitude and position control using graph Laplacians. In: ASME Dynamic Systems and Control Conference; 13–15 September; Cambridge, Massachusetts. ASME; 2010. p. 377-383. DOI: 10.1115/DSCC2010-4036

[11] Hwang I, Tomlin C. Protocol-based conflict resolution for finite information horizon. In: IEEE ACC; 8–10 May; Anchorage, Alaska. IEEExplore; 2002. p. 748-753. DOI: 10.1109/ACC.2002.1024903

[12] Peng L, Zhao Y, Tian B, Zhang J, Bing-Hong W, Hai-Tao Z, Zhou T. Consensus of self-driven agents with avoidance of collisions. Physical Review. 2009;**79**(E). DOI: 10.1103/PhysRevE.79.026113

[13] Olfati-Saber R. Flocking for multi-agent dynamic systems: Algorithms and theory. IEEE Transactions on Automatic Control. 2006;**51**(3):401-420. DOI: 10.1109/TAC.2005.864190

[14] Okoloko I. Multi-Path Planning and Multi-Body Constrained Attitude Control [Dissertation]. Stellenbosch, South Africa: PhD Thesis: Stellenbosch University; 2012. p. 185. Available from: http://hdl handle.net/10019.1/71905

[15] Fax AJ. Optimal and Cooperative Control of Vehicle Formations [Thesis]. Pasadena, CA: PhD Thesis, CALTECH; 2002. p. 135. Available from: thesis.library.caltech.edu/4230/1/Fax_ja_2002.pdf

[16] Kim Y, Mesbahi M. Quadratically constrained attitude control via semidefinite programming. IEEE Transactions on Automatic Control. 2004;**49**:731-735. DOI: 10.1109/TAC.2004.825959

[17] Boyd S, Ghaoui LE, Feron E, Balakrishnan V. Linear Matrix Inequalities in System and Control Theory. Philadelphia, PA: SIAM; 1994. p. 205. ISBN: 0-89871-334-X