Collision-Free 3D Flocking Using the Distributed Simplex Architecture

Usama Mehmood[1], Scott D. Stoller[1], Radu Grosu[2], and Scott A. Smolka[1]

[1] Department of Computer Science, Stony Brook University, USA
[2] Department of Computer Engineering, Technische Universität Wien, Austria

Abstract. The *Distributed Simplex Architecture* (DSA) extends the Simplex control architecture of Sha et al. to provide runtime safety assurance for multi-agent systems under distributed control. In this paper, we show how DSA can be used to ensure collision-free 3D flocking behavior, such that agents avoid colliding with each other and with cuboid-shaped obstacles.

Keywords: Runtime Assurance · Simplex Architecture · Control Barrier Functions · Flocking · Collision Avoidance

1 Introduction

The *Distributed Simplex Architecture* (DSA), is a new runtime assurance technique that provides safety guarantees for multi-agent systems (MASs) under distributed control [1]. DSA is inspired by Sha et al.'s Simplex Architecture [2, 3], but differs from it in significant ways. The Simplex Architecture provides runtime assurance of safety by switching control from an unverified (hence potentially unsafe) *advanced controller* (AC) to a verified-safe *baseline controller* (BC), if the action produced by the AC could result in a safety violation in the near future. The switching logic is implemented in a verified *decision module* (DM).

The applicability of the traditional Simplex architecture is limited to systems with a centralized control architecture, or to those under decentralized control to ensure a "local" safety property that does not depend on the outputs of other controllers. DSA addresses this limitation by re-engineering the traditional Simplex architecture to widen its scope to include MASs. Also, as in [4], it implements *reverse switching* by reverting control back to the AC when it is safe to do so.

This paper provides a brief overview of DSA and then presents a significant DSA application: collision-free 3D flocking, where agents form a flock and navigate through an obstacle field to reach a target location without colliding with each other nor with cuboid-shaped obstacles. Fig. 1 highlights some of the key findings of the case study, showing how a flock of eight agents, initially positioned near the origin, is able to safely navigate around a cuboid to reach a target location. In particular, our results show that DSA prevents all potential collisions. A much simpler version of this case study (2D, no obstacles, no target location) was considered in [1].

We conducted this case study in conjunction with the Klaus Havelund Festschrift. Klaus is an esteemed colleague and friend, and a pioneer in the runtime verification community. We are honored to contribute to the proceedings.

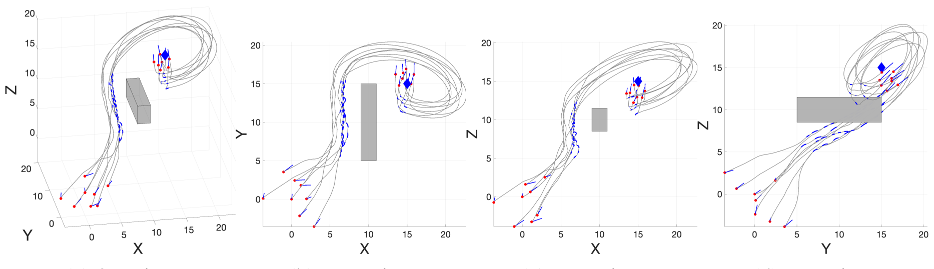

(a) 3D view (b) X-Y view (c) X-Z view (d) Y-Z view

Fig. 1: DSA ensures inter-agent collision avoidance and obstacle avoidance. A flock of eight agents, initialized near the origin, is able to safely navigate to the target location shown as a blue diamond. We represent initial and final positions as red dots, velocities as blue lines, and the trajectory segments where the AC/BC is in control are shown in grey/blue. They grey cuboid is the obstacle.

2 Distributed Simplex Architecture

This section provides a brief overview of the Distributed Simplex Architecture (DSA). For further details please refer to [1]. We formally introduce the MAS safety problem and then briefly discuss the main components of DSA, namely, the distributed baseline controller (BC) and the distributed decision module (DM).

Consider a MAS consisting of k homogeneous agents, denoted as $\mathcal{M} = \{1, ..., k\}$, where the nonlinear control affine dynamics for the i^{th} agent are:

$$\dot{x}_i = f(x_i) + g(x_i)u_i \tag{1}$$

where $x_i \in D \subset \mathbb{R}^n$ is the state of agent i and $u_i \in U \subset \mathbb{R}^m$ is its control input. For an agent i, we define the set of its *neighbors* $\mathcal{N}_i \subseteq \mathcal{M}$ as the agents whose state is accessible to i either through sensing or communication. We denote a combined state of all of the agents in the MAS as the vector $\mathbf{x} = \{x_1^T, x_2^T, ...x_k^T\}^T$ and denote a state of the neighbors of agent i (including i itself) as $x_{\mathcal{N}_i}$. DSA uses discrete-time control: the DMs and controllers execute every η seconds. We assume that all agents execute their DM and controllers simultaneously; this assumption simplifies the analysis.

The set of admissible states $\mathcal{A} \subset \mathbb{R}^{kn}$ consists of all states that satisfy the safety constraints. A constraint $C : D^k \to \mathbb{R}$ is a function from k-agent MAS states to the reals. In this paper, we are primarily concerned with *binary constraints* (between neighboring agents) of the form $C_{ij} : D \times D \to \mathbb{R}$, and *unary constraints* of the form

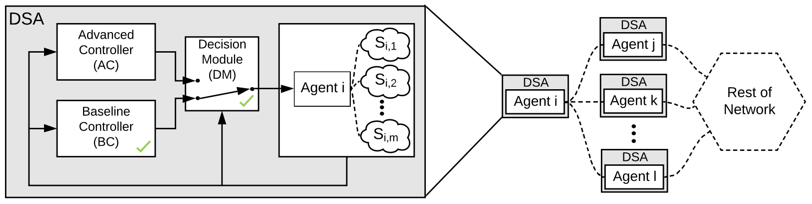


Fig. 2: DSA for the MAS on the right. Agents are homogeneous and operate under DSA; the figure zooms in on DSA components for agent i. Sensed state of agent i's j^{th} neighbor is denoted as $S_{i,j}$. AC, BC, and DM take as input the state of the agent and its neighbors.

$C_i : D \to \mathbb{R}$. Hence, the set of admissible states, $\mathcal{A} \subset \mathbb{R}^{kn}$ are the MAS states of $\mathbf{x} \in \mathbb{R}^{kn}$ such that all of the unary and binary constraints are satisfied.

Formally, DSA is solving the following problem. Given a MAS defined as in Eq. (1) and $\mathbf{x}(0) \in \mathcal{A}$, design a BC and DM to be used by all agents such that the MAS remains safe; i.e. $\mathbf{x}(t) \in \mathcal{A}$, $\forall\, t > 0$.

In DSA, illustrated in Fig. 2, for each agent, there is a verified-safe BC and a verified switching logic such that if all agents operate under DSA, then safety of the MAS is guaranteed. The BC and DM along with the AC are distributed and depend only on local information. DSA itself is *distributed* in that it involves one local instance of traditional Simplex per agent such that the conjunction of their respective safety properties yields the desired safety property for the entire MAS. For example, consider our flocking case study, where we want to establish collision-freedom for the entire MAS. This can be accomplished in a distributed manner by showing that each local instance of Simplex, say for agent i, ensures collision-freedom for agent i and its neighboring agents. Moreover, DSA allows agents to switch their mode of operation independently. At any given time, some agents may be operating in AC mode while others are operating in BC mode.

2.1 Baseline Controller

Our approach to the design of the BC and DM leverages *Control Barrier Functions* (CBFs), which have been used to synthesize safe controllers [5, 6, 7], and are closely related to Barrier Certificates used for safety verification of closed dynamical systems [8, 9]. A CBF is a mapping from the system's (i.e., plant's) state space to a real number, with its zero level-set partitioning the state space into safe and unsafe regions. If certain inequalities on the derivative of the CBF in the direction of the state trajectories (also known as the Lie derivative) are satisfied, then the corresponding control actions are considered safe (admissible). For binary safety constraints, the corresponding inequalities on the Lie derivative of the CBF are conditions on the control actions of a pair of agents. The distributed control of the two agents cannot independently satisfy the binary constraint without running an agreement protocol.

In accordance with [7], we solve the problem of the satisfaction of binary constraints by partitioning a binary constraint into two unary constraints such that the satisfaction of the unary constraints by agents i and j implies the satisfaction of the binary constraint (but not necessarily vice versa).

$$\left.\begin{aligned} P_{ij}u_i \leq b_{ij}/2 \\ Q_{ij}u_j \leq b_{ij}/2 \end{aligned}\right\} \Rightarrow \begin{bmatrix} P_{ij} \; Q_{ij} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} \leq b_{ij} \tag{2}$$

In DSA, the BC is designed as an optimal controller with the goal of increasing a utility function based on the Lie derivatives of the CBFs. As CBFs are a measure of the safety of a state, optimizing for control actions with higher Lie derivative values provides a direct way to make the state safer. The safety of the BC is further guaranteed by constraining the control action to remain in a set of admissible actions that satisfy certain inequalities on the Lie derivatives of the CBFs. CBFs are also used in the design of the switching logic, as they provide an efficient method for checking whether an action could lead to a safety violation during the next time step.

2.2 Decision Module

Each agent's DM implements the switching logic for both forward and reverse switching. Control is switched from the AC to the BC if the *forward switching condition* (FSC) is true. Similarly, control is reverted back to the AC (from the BC) if the *reverse switching condition* (RSC) is true.

We derive the switching conditions from the CBFs as follows. To ensure safety, the FSC must be true in a state $x_{\mathcal{N}_i}(t)$ if an unrecoverable state is reachable from $x_{\mathcal{N}_i}(t)$ in one time step η. For a CBF in a given state, we define a *worst-case action* to be an action that minimizes the CBF's Lie derivative. The check for one-step reachability of an unrecoverable state is based on the minimum value of the Lie derivative of the CBFs, which corresponds to the worst-case actions by the agents. Hence, for each CBF h, we define a minimum threshold value $\lambda_h(x_{\mathcal{N}_i})$ equal to the magnitude of the minimum of the Lie derivative of the CBF times η, and we switch to the BC if, in the current state, the value of any CBF h is less than $\lambda_h(x_{\mathcal{N}_i})$. This directly ensures that none of the CBFs can decrease enough to become negative during the next control period.

We derive the RSC using a similar approach, except the inequalities are reversed and an m-time-step reachability check with $m > 1$ is used; the latter prevents frequent back-and-forth switching between the AC and BC. The RSC holds if in the current state, the value of each CBF h is greater than the threshold $m\lambda_h(x_{\mathcal{N}_i})$. This results in an FSC and RSC of the following form:

$$FSC(x_{\mathcal{N}_i}) = (h_i < \lambda_{h_i}(x_{\mathcal{N}_i})) \vee (\exists j \in \mathcal{N}_i \mid h_{ij} < \lambda_{h_{ij}}(x_{\mathcal{N}_i})) \tag{3}$$

$$RSC(x_{\mathcal{N}_i}) = (h_i > m\lambda_{h_i}(x_{\mathcal{N}_i})) \wedge (\forall j \in \mathcal{N}_i \mid h_{ij} > m\lambda_{h_{ij}}(x_{\mathcal{N}_i})) \tag{4}$$

3 Collision-Free Flocking

We evaluate DSA on the distributed flocking problem with the goal of preventing inter-agent collisions and collisions with stationary cuboid-shaped obstacles. Consider a MAS consisting of k robotic agents with double integrator dynamics, indexed by $\mathcal{M} = \{1, \ldots, k\}$:

$$\begin{bmatrix} \dot{p}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} 0 & I_{3\times3} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_i \\ v_i \end{bmatrix} + \begin{bmatrix} 0 \\ I_{3\times3} \end{bmatrix} a_i \tag{5}$$

where p_i, v_i, $a_i \in \mathbb{R}^3$ are the position, velocity and acceleration of agent $i \in \mathcal{M}$, respectively. The magnitudes of velocities and accelerations are bounded by $\bar{v}$ and $\bar{a}$, respectively. Acceleration a_i is the control input for agent i. There are l static cuboid-shaped obstacles, indexed by $\mathcal{O} = \{1, ..., l\}$.

As DSA is a discrete-time protocol, the state of the DM and the a_i's are updated every η seconds. The *state* of an agent i is denoted by the vector $s_i = [p_i^T v_i^T]^T$. The *state* of the entire flock at time t is denoted by the vector $\mathbf{s}(t) = [\mathbf{p}(t)^T\ \mathbf{v}(t)^T]^T \in \mathbb{R}^{6k}$, where $\mathbf{p}(t) = [p_1^T(t) \cdots p_k^T(t)]^T$ and $\mathbf{v}(t) = [v_1^T(t) \cdots v_k^T(t)]^T$ are the vectors respectively denoting the positions and velocities of the flock at time t. For ease of notation, we sometimes use $\mathbf{s}$ and $\mathbf{s}_i$ to refer to the state variables $\mathbf{s}(t)$ and $\mathbf{s}_i(t)$, respectively, without the time index.

We assume that an agent can accurately sense the positions and velocities of objects in a sphere of radius r. The sensed objects include the other agents and the static cuboid-shaped obstacles. The set of *spatial neighbors* of agent i is defined as $\mathcal{N}_i(\mathbf{p}) = \{j \in \mathcal{M} \mid j \neq i \wedge \|p_i - p_j\| < r\}$, where $\|\cdot\|$ denotes the Euclidean norm. The obstacles which are completely or partially within the sensing range of agent i are denoted by the set $\mathcal{O}_i$.

The MAS is characterized by a set of operational constraints which include physical limits and safety properties. States that satisfy the operational constraints are called *admissible*, and are denoted by the set $\mathcal{A} \in \mathbb{R}^{6k}$. The desired safety property is that no agent is in a "state of collision" with any other agent or any obstacle. A pair of agents is considered to be in a *state of collision* if the Euclidean distance between them is less than a threshold distance $d_\alpha \in \mathbb{R}^+$, resulting in binary safety constraints of the form: $\|p_i - p_j\| - d_\alpha \geq 0\ \forall\ i \in \mathcal{M}, j \in \mathcal{N}_i$. Similarly, an agent is considered to be in a state of collision with an obstacle if the shortest Euclidean distance between the agent and the obstacle is less than a threshold distance $d_\beta \in \mathbb{R}^+$, resulting in unary safety constraints of the form: $\|p_i - p_o\| - d_\beta \geq 0\ \forall\, i \in \mathcal{M}, o \in \mathcal{O}$. A state $\mathbf{s}$ is $recoverable$ if all agents can brake (de-accelerate) relative to each other and relative to the stationary obstacles without colliding. Otherwise, $\mathbf{s}$ is considered *unrecoverable*.

3.1 Synthesis of Control Barrier Function

For an agent i, CBFs are defined for all its neighboring agents and for all the obstacles in its sensing range. We assume the following two conditions on the sensing range:

$$\begin{aligned} r &> \bar{v}\eta + \frac{\bar{v}^2}{4\bar{a}} + d_\alpha \\ r &> \bar{v}\eta + \frac{\bar{v}^2}{2\bar{a}} + d_\beta \end{aligned} \tag{6}$$

These conditions ensure collision freedom, during the next decision period η, with the agents and obstacles outside the sensing range, respectively. Hence CBFs are not needed for objects outside the sensing range.

For each agent i, the local admissible set $\mathcal{A}_i \subset \mathbb{R}^6$ is the set of states $s_i \in \mathbb{R}^6$ which satisfy all the unary obstacle avoidance constraints. The set $\mathcal{S}_i \subset \mathcal{A}_i$ is defined as the super-level set of the CBF $h_i : \mathbb{R}^6 \to \mathbb{R}$, which is designed to ensure forward-invariance of $\mathcal{A}_i$. Similarly, for a pair of neighboring agents i, j where $i \in \mathcal{M}, j \in \mathcal{N}_i$, the pairwise admissible set $\mathcal{A}_{ij} \subset \mathbb{R}^{12}$ is the set of pairs of states which satisfy all the binary inter-agent collision avoidance constraints. The set $\mathcal{S}_{ij} \subset \mathcal{A}_{ij}$ is defined as the super-level set of the CBF $h_{ij} : \mathbb{R}^{12} \to \mathbb{R}$ designed to ensure forward-invariance of $\mathcal{A}_{ij}$. The recoverable set $\mathcal{R}_{ij} \subset \mathbb{R}^{12}$, for a pair of neighboring agents i, j where $i \in \mathcal{M}, j \in \mathcal{N}_i$, is defined in terms of $\mathcal{S}_i$, $\mathcal{S}_j$ and $\mathcal{S}_{ij}$.

$$\mathcal{S}_i = \{s_i \in \mathbb{R}^6 | h_i(s_i) \geq 0\} \tag{7}$$

$$\mathcal{S}_{ij} = \{(s_i, s_j) \in \mathbb{R}^{12} | h_{ij}(s_i, s_j) \geq 0\} \tag{8}$$

$$\mathcal{R}_{ij} = (\mathcal{S}_i \times \mathcal{S}_j) \cap \mathcal{S}_{ij} \tag{9}$$

The recoverable set $\mathcal{R} \subset \mathcal{A}$ for the entire MAS is defined as the set of system states in which $(s_i, s_j) \in \mathcal{R}_{ij}$ for every pair of agents i, j.

In accordance with [10], the inter-agent collision avoidance CBF function $h_{ij}(s_i, s_j)$ is based on a safety constraint over a pair of neighboring agents i, j. The safety constraint ensures that for any pair of agents, the maximum deceleration can always keep the agents at a distance greater than d_α from each other. As introduced earlier, d_α is the threshold distance that defines an inter-agent collision. Considering that the tangential component of the relative velocity, denoted by Δv, causes a collision, the constraint regulates Δv by application of maximum acceleration to reduce Δv to zero. Hence, the safety constraint can be represented as the following condition on the inter-agent distance $\|\Delta \mathbf{p}_{ij}\| = \|p_i - p_j\|$, the stopping distance $(\Delta v)^2/4\bar{a}$, and the safety threshold distance d_α:

$$\|\Delta \mathbf{p}_{ij}\| - \frac{(\Delta v)^2}{4\bar{a}} \geq d_\alpha \tag{10}$$

$$h_{ij}(s_i, s_j) = \sqrt{4\bar{a}(\|\Delta \mathbf{p}_{ij}\| - d_\alpha)} - \Delta v \geq 0 \tag{11}$$

The stopping distance is the distance covered while the relative speed reduces from Δv to zero under a deceleration of $2\bar{a}$. As introduced earlier, $\bar{a}$ is the upper limit on the magnitude of accelerations for all agents. The constraint in Eq. (10) is re-arranged to get the CBF h_{ij} given in Eq. (11). Similarly, the obstacle avoidance CBF $h_i^{(j)}(s_i)$ is defined for the agent i and (stationary) obstacle j where the obstacle is within the sensing range of the agent:

$$h_i^{(j)}(s_i) = \sqrt{2\bar{a}(\|O_{ij}\| - d_\beta)} - v_i^T \frac{O_{ij}}{\|O_{ij}\|} \geq 0 \tag{12}$$

where $\|O_{ij}\|$ is the shortest distance between the agent i and the obstacle j. The vector $O_{ij} = p_i - o_j^{(i)}$, where $o_j^{(i)}$ is the point on obstacle j closest to the agent i. The zero-level set of $h_i^{(j)}$ separates the states from which the agent can brake to a stop, maintaining a distance of at least d_β from the obstacle j.

The admissible control space for the BC is defined by constraining the Lie derivatives of the CBFs. For the CBF h_{ij}, linear constraint on the accelerations for agents i and j, are obtained by constraining the Lie derivative of the CBF h_{ij} to be greater than $-\alpha(h_{ij})$, where $\alpha : \mathbb{R} \to \mathbb{R}$ is an extended class $\mathcal{K}$ function i.e., strictly increasing and $\alpha(0) = 0$. We set $\alpha(h_{ij}) = \gamma h_{ij}^3$, as in [10], where $\gamma \in \mathbb{R}^+$, resulting in the following constraint on the accelerations of agents i, j:

$$\begin{aligned}\frac{\Delta\mathbf{p}_{ij}^T(\Delta\mathbf{a}_{ij})}{\|\Delta\mathbf{p}_{ij}\|} - \frac{(\Delta\mathbf{v}_{ij}^T\Delta\mathbf{p}_{ij})^2}{\|\Delta\mathbf{p}_{ij}\|^3} + \frac{\|\Delta\mathbf{v}_{ij}\|^2}{\|\Delta\mathbf{p}_{ij}\|}& \\ + \frac{2\bar{a}\Delta\mathbf{v}_{ij}^T\Delta\mathbf{p}_{ij}}{\|\Delta\mathbf{p}_{ij}\|\sqrt{4\bar{a}(\|\Delta\mathbf{p}_{ij}\| - d_\alpha)}} &\geq -\gamma h_{ij}^3\end{aligned} \tag{13}$$

where the left-hand side is the Lie derivative of the CBF h_{ij} and $\Delta\mathbf{p}_{ij} = p_i - p_j$, $\Delta\mathbf{v}_{ij} = v_i - v_j$, and $\Delta\mathbf{a}_{ij} = a_i - a_j$ are the vectors representing the relative position, the relative velocity, and the relative acceleration of agents i and j, respectively. We further note that the binary constraint (13) can be reformulated as $\left[\begin{smallmatrix} P_{ij} & Q_{ij}\end{smallmatrix}\right]\left[\begin{smallmatrix} a_i \\ a_j\end{smallmatrix}\right] \leq b_{ij}$, and hence can be split into two unary constraints $P_{ij}u_i \leq b_{ij}/2$ and $Q_{ij}u_j \leq b_{ij}/2$, following the convention in Eq. (2). Constraints similar to Eq. (13) are also computed for the obstacle avoidance CBF $h_i^{(j)}$ which can be denoted as $B_i^{(j)}u_i \leq c_i^{(j)}$. The set of safe accelerations for an agent i, denoted by $\mathcal{K}_i(\mathbf{s}_i) \subset \mathbb{R}^3$, is defined as the intersection of the half-planes defined by the Lie-derivative-based constraints, where each neighboring agent and each obstacle within the sensing range contributes a single constraint:

$$\mathcal{K}_i(\mathbf{s}_i) = \left\{a_i \in \mathbb{R}^2 \mid P_{ij}u_i \leq b_{ij}/2 \; \forall j \in \mathcal{N}_i \wedge B_i^{(j)}u_i \leq c_i^{(j)} \; \forall j \in \mathcal{O}_i\right\} \tag{14}$$

With the CBFs for collision-free flocking defined in (11) and (12) and the admissible control space defined in (14), the FSC, and RSC follow from (3), and (4), respectively. The BC is designed as a constrained optimal controller as defined in Section 2.1.

3.2 Advanced Controller

We extend the Reynolds flocking model [11] to include target seeking behaviour and use it as the AC. In the Reynolds model, the acceleration a_i for an agent is a weighted sum of three acceleration terms based on simple rules of interaction with neighboring agents: *separation* (move away from your close-by neighbors), *cohesion* (move towards the centroid of your neighbors), and *alignment* (match your velocity with the average velocity of your neighbors). We add two more terms for the target seeking behaviour: A *goto target* term which forces the agent to move towards the fixed target location, and a *velocity damping term*, which brings the agent to a stop at the target location, preventing it from overshooting the target and oscillating about it. The mathematical expressions for the acceleration terms are

$$
\begin{aligned}
a_i^s &= \frac{1}{|\mathcal{N}i|} \cdot \left(\sum_{j \in \mathcal{N}i} \frac{p_i - p_j}{\|p_i - p_j\|^2} \right) \\
a_i^c &= \left(\frac{1}{|\mathcal{N}i|} \cdot \sum_{j \in \mathcal{N}i} p_j \right) - p_i \\
a_i^{al} &= \left(\frac{1}{|\mathcal{N}_i|} \cdot \sum_{j \in \mathcal{N}_i} v_j \right) - v_i \\
a_i^{gt} &= \frac{t - p_i}{\|t - p_i\|} \\
a_i^{dp} &= -\frac{v_i}{\|t - p_i\|^2}
\end{aligned}
\qquad (15)
$$

where $t \in \mathbb{R}^3$ is the target location and $a_i^s, a_i^c, a_i^{al}, a_i^{gt}, a_i^{dp} \in \mathbb{R}^3$ are the acceleration terms corresponding to separation, cohesion, alignment, goto target, and velocity damping, respectively.

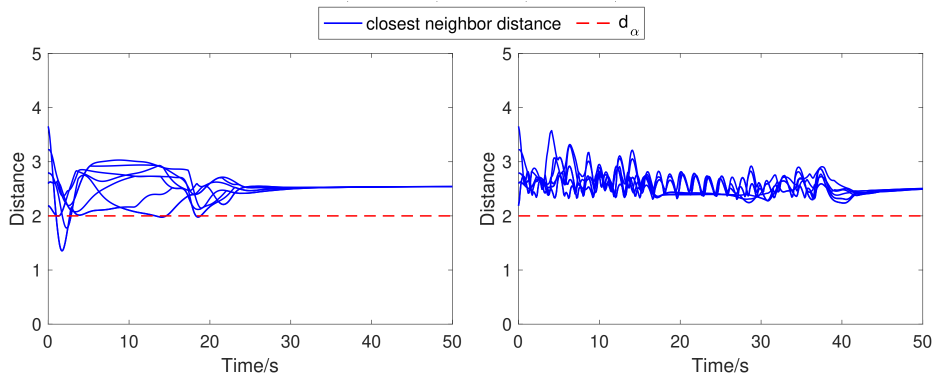
The acceleration for agent i is $a_i = w_s a_i^s + w_c a_i^c + w_{al} a_i^{al} + w_{gt} a_i^{gt} + w_{dp} a_i^{dp}$, where $w_s, w_c, w_{al}, w_{gt}, w_{dp} \in \mathbb{R}^+$ are scalar weights. We note that the Reynolds model does not guarantee collision avoidance; see Fig 3(a). Nevertheless, when the flock stabilizes, the average distance between an agent and its closest neighbors is determined by the weights of the interaction terms.

3.3 Experimental Results

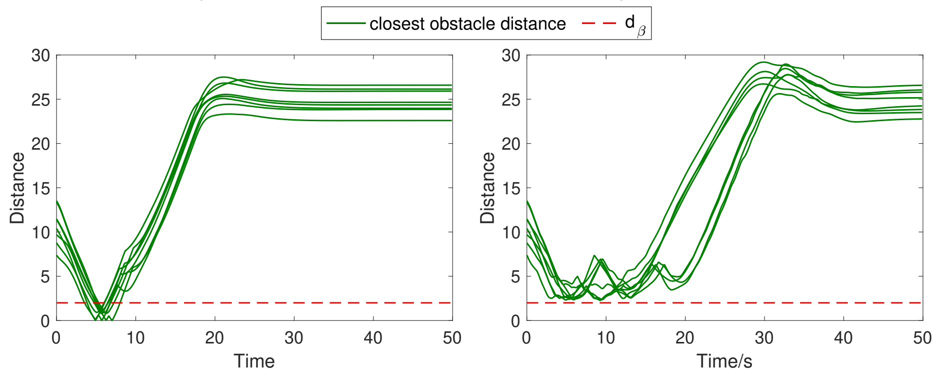
The number of agents in the MAS is $k = 8$. The other parameters used in the experiments are $r = 4$, $\bar{a} = 5$, $\bar{v} = 2.5$, $d_\alpha = d_\beta = 2$, $\eta = 0.2$s, $\gamma = 0.5$, and $m = 2$. There are three cuboid-shaped obstacles in the path to the target location. The length of the simulations is 50 seconds. The initial positions and the initial velocities are uniformly sampled from $[-10, 10]^2$ and $[0, 1]^2$, respectively, and we ensure that the initial state is recoverable. The weights of the Reynolds model terms are chosen experimentally to ensure that no pair of agents are in a state of collision in the steady state. They are set to $w_s = 3.0$, $w_c = 1.5$, $w_{al} = 1.2$, $w_t = 0.3$, and $w_{dp} = 3.0$.

We performed two simulations, starting from the same initial configuration. In the first simulation, Reynolds model is used to control all of the agents for the duration of the simulation. In the second simulation, Reynolds model is wrapped with a verified safe BC and DM designed using DSA.

To recall, the safety property is that agents maintain a distance greater than d_α from each other and distance greater than d_β from the obstacles. Figs. 3(a) and (b) plot, for the duration of the simulations, the distance between each agent and its closest neighbor. As evident from Fig. 3(a), Reynolds model results in safety violations. In contrast, as shown in Fig. 3(b), DSA preserves safety, maintaining a separation greater than d_α between all agents. Figs. 3(c) and 3(d) plot, for the duration of the simulations, the distance



(a) Reynolds Model (b) Reynolds Model with DSA



(c) Reynolds Model (d) Reynolds Model with DSA

Fig. 3: DSA is able to avoid all inter-agent collisions and obstacle collisions for a flock of eight agents.

between each agent and the closest point on the closest obstacle. As Reynolds model is not designed for obstacle avoidance, it results in a number of agent-obstacle collisions, whereas DSA completely prevents them. We further observe that the average time the agents spent in BC mode is only 7.8 percent of the total duration of the simulation, demonstrating that for this case study, DSA is largely non-invasive.

4 Conclusion

The Distributed Simplex Architecture is a runtime safety assurance technique for multi-agent systems. DSA is distributed in the sense that it involves one local instance of traditional Simplex per agent such that the conjunction of their respective safety properties yields the desired safety property for the entire MAS. In this paper, we have demonstrated the effectiveness of the DSA approach by successfully applying it to collision-free 3D flocking.

Bibliography

[1] U. Mehmood, S. D. Stoller, R. Grosu, S. Roy, A. Damare, and S. A. Smolka, “A distributed simplex architecture for multi-agent systems,” *CoRR*, vol. abs/2012.10153, 2020. [Online]. Available: https://arxiv.org/abs/2012.10153

[2] D. Seto and L. Sha, “A case study on analytical analysis of the inverted pendulum real-time control system,” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-99-TR-023, 1999.

[3] L. Sha, “Using simplicity to control complexity,” *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[4] D. Phan, R. Grosu, N. Jansen, N. Paoletti, S. A. Smolka, and S. D. Stoller, “Neural simplex architecture,” in *Proceedings of NASA Formal Methods Symposium (NFM 2020)*, 2020.

[5] T. Gurriet, A. Singletary, J. Reher, L. Ciarletta, E. Feron, and A. Ames, “Towards a framework for realizable safety critical control through active set invariance,” in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, 2018, pp. 98–106.

[6] M. Egerstedt, J. N. Pauli, G. Notomista, and S. Hutchinson, “Robot ecology: Constraint-based control design for long duration autonomy,” *Annual Reviews in Control*, vol. 46, pp. 1 – 7, 2018.

[7] L. Wang, A. D. Ames, and M. Egerstedt, “Safety barrier certificates for heterogeneous multi-robot systems,” in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 5213–5218.

[8] S. Prajna and A. Jadbabaie, “Safety verification of hybrid systems using barrier certificates,” in *Hybrid Systems: Computation and Control, 7th International Workshop*, ser. Lecture Notes in Computer Science, R. Alur and G. J. Pappas, Eds., vol. 2993. Springer, 2004, pp. 477–492.

[9] S. Prajna, “Barrier certificates for nonlinear model validation,” *Autom.*, vol. 42, no. 1, pp. 117–126, 2006.

[10] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, “Control barrier certificates for safe swarm behavior,” in *ADHS*, ser. IFAC-PapersOnLine, M. Egerstedt and Y. Wardi, Eds., vol. 48, no. 27. Elsevier, 2015, pp. 68–73.

[11] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 25–34, Aug. 1987.