Robust MADER: Decentralized and Asynchronous Multiagent Trajectory Planner Robust to Communication Delay

Kota Kondo, Jesus Tordesillas, Reinaldo Figueroa, Juan Rached, Joseph Merkel, Parker C. Lusk, and Jonathan P. How

Abstract—Although communication delays can disrupt multiagent systems, most of the existing multiagent trajectory planners lack a strategy to address this issue. State-of-the-art approaches typically assume perfect communication environments, which is hardly realistic in real-world experiments. This paper presents Robust MADER (RMADER), a decentralized and asynchronous multiagent trajectory planner that can handle communication delays among agents. By broadcasting both the newly optimized trajectory and the committed trajectory, and by performing a delay check step, RMADER is able to guarantee safety even under communication delay. RMADER was validated through extensive simulation and hardware flight experiments and achieved a 100% success rate of collision-free trajectory generation, outperforming state-of-the-art approaches.

Supplementary Material

Video: https://youtu.be/vH09kWJOBYs
Code: https://github.com/mit-acl/rmader

I. INTRODUCTION

Multiagent UAV trajectory planning has been extensively studied in the literature for its wide range of applications. These planners can be centralized [1], [2], [3] (one machine plans every agent’s trajectory) or decentralized [4], [5], [6] (each agent plans its own trajectory). Decentralized planners are more scalable and robust to failures of the centralized machine. Despite these advantages, a decentralized scheme requires communication between the agents, and communication delays could potentially introduce failure in the trajectory deconfliction between the agents, which is essential to guarantee safety [7]. Multiagent planners can also be classified according to whether or not they are asynchronous. In an asynchronous setting, each agent independently triggers the planning step without considering the planning status of other agents. Asynchronous approaches do not require a synchronous mechanism among agents and therefore more scalable than synchronous approaches, but they are also more susceptible to communication delays since agents are planning and executing trajectories independently.

Many decentralized state-of-the-art trajectory planners do not consider communication delays or explicitly state assumptions about communication. For example, the planners presented in SCP [8], decNS [9], and LSC [10] are decentralized and synchronous, but SCP and decNS implicitly and LSC explicitly assume a perfect communication environment without any communication delays.

Aerospace Controls Laboratory, MIT, 77 Massachusetts Ave, Cambridge, MA, USA {kkondo, jtorde, reyfp, jrached, jmerkel, plusk, jhow}@mit.edu

1As in [4], we define asynchronous planning to be when the agent triggers trajectory planning independently without considering the planning status of other agents. However, ADPP [13] implements a prioritized asynchronous approach, meaning plannings are not fully independently triggered.
TABLE I. State-of-the-art Decentralized Multiagent Planners.

| Method             | Asynchronous? | Handles Comm. Delay? | Hardware Demonstration |
|--------------------|---------------|----------------------|------------------------|
| decNS [9]          | No            | No                   | No                     |
| SCP [8]            | No            | No                   | Yes                    |
| LSC [10]           | No            | No                   | Yes                    |
| decMPC [11]        | No            | Yes                  | No                     |
| decGroup [12]      | Yes/No²       | No                   | Yes                    |
| ADPP [13]          | Yes³          | No                   | Yes                    |
| MADER [4]          | Yes           | No                   | No                     |
| EGO-Swarm [5]      | Yes           | No                   | Yes                    |
| AsyncBVC [14]      | Yes           | Yes                  | Yes                    |
| RMADER (proposed)  | Yes           | Yes                  | Yes                    |

² decGroup triggers joint-optimization in dense environments and switches to a centralized, synchronous planner.
³ Asynchronous but requires priority information for planning.

TABLE II. Definitions of the different delay quantities: Note that, by definition, $0 \leq \delta_{\text{introd}} \leq \delta_{\text{act}} \leq \delta_{\max}$.

| Symbol | Description |
|--------|-------------|
| $\delta_{\text{act}}$ | Actual communication delays among agents. |
| $\delta_{\max}$ | Possible maximum communication delay. |
| $\delta_{\text{introd}}$ | Introduced communication delay in simulations. |
| $\delta_{\text{DC}}$ | Length of Delay Check in RMADER. To guarantee safety, $\delta_{\max} \leq \delta_{\text{DC}}$ must be satisfied. |

3) Extensive set of decentralized hardware experiments using 6 UAVs, and achieving velocities up to 2.8 m/s.

II. TRAJECTORY DECONFLICTION

In MADER [4] and RMADER, UAV's plan trajectories asynchronously and broadcast the results to each other. Each agent uses these trajectories as constraints in the optimization problem. Assuming no communication delays exist, safety can be guaranteed using our previous approach presented in MADER (summarized in Section II-A). This safety guarantee, however, breaks when an agent’s planned trajectory is received by other agents with some latency. Section II-B shows how RMADER guarantees safety even with communication delays. We use the definitions shown in Table II.

A. MADER Deconfliction

MADER [4] guarantees collision-free trajectories under ideal communication through the use of the planning stages shown in Fig. 2. An agent plans its initial trajectory during Optimization (O), followed by Check (C) to ensure its plan does not lead to a collision. Finally, Recheck (R) is used to check if the agent received any trajectory updates from other agents during C - if so, an agent starts over planning at O. Although MADER does not have explicit safety guarantees in the presence of communication delays, its trajectories are still collision free for cases 1 and 2 shown in Fig. 2. However, collisions may occur in cases 3 and 4 of Fig. 2. These four cases are summarized in Fig. 2 and Table III.

B. Robust MADER Deconfliction

To achieve robustness to communication delays, we replace the Recheck with Delay Check (DC), where each agent repeatedly checks if its newly optimized trajectory conflicts with other agents’ trajectories. If an agent detects conflicts during DC, it discards the new trajectory and starts another O while executing its previous trajectory. If no collisions are detected in DC, it starts executing the new trajectory. To guarantee collision-free trajectory generation, DC needs to be longer than the possible longest communication delay (i.e., $\delta_{\text{DC}} \geq \delta_{\max}$). That way, an agent can always keep at least one collision-free trajectory. It could, however, not be ideal for introducing such a long $\delta_{\text{DC}}$, and therefore, in Section III we also tried $\delta_{\text{DC}} < \delta_{\max}$ and measure its performance. Fig. 3 shows how RMADER
TABLE III. Safety guarantees under communication delays: Depending on when trajA is received by Agent B, the deconfliction takes place at different stages. MADER does not guarantee safety if trajA is received during $R_B$ or during the following iteration, while RMADER guarantees safety in all the cases. Note that in RMADER, if Agent B does not receive trajA by the end of $D_C$, then the deconfliction is performed by Agent A (specifically, in $C_A$ or $D_C$) and not by Agent B. Agent A will use trajBnew and/or trajB for this.

TABLE IV. Differences between MADER and RMADER

| MADER | RMADER |
|--------------------|-----------------------|
| Upon successful C and R, the newly optimized trajectory is broadcast to other agents | Upon successful C, trajBnew is broadcast. After DC, the committed trajectory traj (which is either trajAprev or trajA, depending on whether DC is satisfied or not) is broadcast |
| DC is a sequence of collision checks | DC lasts $\delta_D$ seconds |
| R is a Boolean check to see if the agent received traj in C | R is very short |

Algorithm 1 Robust MADER - Agent B

Require: trajB, a feasible trajectory

1: while not goal reached do
2: trajBnew = Optimization()
3: if Check(trajBnew) == False then
4: Go to Line 2
5: end if
6: Broadcast trajBnew
7: if Delay Check(trajBnew) == False then
8: trajB ← trajBprev and go to Line 11
9: end if
10: trajB ← trajBnew
11: Broadcast trajB
12: end while

Algorithm 2 Delay Check - Agent B

1: function Delay Check(trajBnew)
2: for $\delta_D$ seconds do
3: if trajBnew collides with any trajectory in $Q_B$ then
4: return False
5: end if
6: end for
7: return True
8: end function
Fig. 4. Agent B stores in $Q_B$ the last committed trajectory of Agent A. It will also contain the newly optimized trajectory $\text{traj}_{A_{new}}$ while the new committed trajectory has still not been received by Agent B.

Fig. 5. 10 agents employing RMADER exchange their positions in a circle configuration.

(a) For Agent J, the colored trajectory is the committed (safety-guaranteed) trajectory ($\text{traj}_j$), and the grey trajectory is the newly optimized trajectory ($\text{traj}_{\text{new}}$).

(b) Actual trajectories flown by the agents. All 10 agents successfully swap their positions in a circle configuration.

III. SIMULATION RESULTS

We tested Slow EGO-Swarm, EGO-Swarm [5], MADER [4], and RMADER (proposed) on a general-purpose N2 Google Cloud instance with 32 Intel Core i7s. In each scenario, we conducted 100 simulations with 10 agents positioned in a 10 m radius circle, exchanging positions diagonally as shown in Fig. 5. Note that this paper convexified MADER optimization problem as detailed in Appendix I, and we used the convex optimization problem for both MADER and RMADER. The maximum dynamic limits (velocity, acceleration, and jerk) for these algorithms are set to 10 m/s, 20 m/s$^2$, and 30 m/s$^3$. EGO-Swarm carries out a sequential startup - agents commits their first trajectory in a pre-determined order to avoid unnecessary trajectory conflicts. We also introduced 0.25 s-apart startup into MADER and RMADER.

Slow EGO-Swarm is EGO-Swarm with smaller dynamic limits. We first tested EGO-Swarm with default parameters provided in [5] and saw a significant number of conflicts. Therefore we increased the weights of the collision costs in EGO-Swarm’s cost function up to 1000 (we tried more than 1000, but it did not change the results) while other weights (s.t. trajectory feasibility) are on the order of single digits; however, we still observed collisions (as seen in the second row of Table V). We thus decreased the maximum velocity and acceleration of EGO-Swarm down to 5 m/s and 10 m/s$^2$, which we define as Slow EGO-Swarm.

Although we introduced a fixed $\delta_{\text{intro}}$ for simulated communications, $\delta_{\text{actual}}$ can be larger due to the simulation computer’s computational limitations. The communication delays observed in simulation are shown in Fig. 6f for five nominal values of $\delta_{\text{intro}}$ ($\delta_{\text{intro}} = 0, 50, 100, 200, \text{and } 300 \text{ ms}$). As long as $\delta_{\text{DC}} \geq \delta_{\text{max}}$ holds, RMADER can generate collision-free trajectories.

Table V and Fig. 6 showcase each approach’s performance in simulations. RMADER was implemented in the case of (1) $\delta_{\text{DC}} \geq \delta_{\text{max}}$ ($\delta_{\text{DC}} > 100$th percentile of $\delta_{\text{actual}}$) and (2) $\delta_{\text{max}} \geq \delta_{\text{DC}}$ ($\delta_{\text{DC}} \approx 75$th percentile of $\delta_{\text{actual}}$). When $\delta_{\text{DC}} \geq \delta_{\text{max}}$ holds, collision-free trajectory planning is guaranteed, and therefore RMADER generates 0 collisions for all the $\delta_{\text{intro}}$, while other approaches suffer collisions. As expected, the longer $\delta_{\text{intro}}$ more collisions Slow EGO-Swarm, EGO-Swarm, and MADER generate. In the case of (2) $\delta_{\text{max}} \geq \delta_{\text{DC}}$, although safety is not theoretically guaranteed, since $\delta_{\text{DC}}$ is long enough, RMADER succeeds to generate collision-free trajectories. Note that Case (2) could have collisions in case agents have conflicted trajectories and their trajectories fall into the rest of $\approx 25\%$.

It is also worth mentioning that RMADER’s robustness to communication delays is obtained by layers of conflict checks and agents periodically occupying two trajectory spaces, which can result in generating conservative trajectories and trading off UAV performance. Avg. Number of Stops and Avg. Number of Stops in Table V, for instance, suggests more stoppage than other approaches. As $\int \|a\|^2 dt$ and $\int \|j\|^2 dt$ show RMADER’s trajectories are less smooth than MADER, and RMADER takes longer Travel Time than others (Slow EGO-Swarm takes more but that is because of its smaller dynamic limits, and therefore a direct comparison is not fair).

IV. HARDWARE EXPERIMENTS

A total of 10 hardware experiments (5 flights for each) demonstrate RMADER’s robustness to communication delays as well as MADER’s shortcomings. Each flight test had 6 UAVs in the 9.2 m x 7.5 m x 2.5 m flight space and lasted $\approx 1 \text{ min}$. All the planning and control run onboard the UAV, and the state estimation is obtained by fusing IMU measurements with an external motion capture system. A safety mechanism running in parallel reports potential collisions and sends commands to the UAVs to avoid colliding.

During the MADER hardware experiments due to the effects of communication delays, 7 potential collisions were detected by the safety mechanism. RMADER, on the other hand, did not generate conflicts. A snapshot of one of the RMADER experiments is shown in Fig. 1a, and a successful trajectory deconfliction despite the communication delay is shown in Fig. 8. The maximum velocities and average flight distances achieved during the MADER and RMADER
TABLE V. Cases δintrod = 0 ms, δintrod = 50 ms, δintrod = 100 ms, (see Fig. 6f for actual message delays). The bold values represent the case where δPC ≥ δmax, which is the necessary condition to ensure safety.

| Method       | δPC [ms] | Collision [%] | Avg number of stops | $\int [m/s]dt$ | $\int [m/s]^2dt$ | Travel Time [s] |
|--------------|----------|---------------|---------------------|----------------|-----------------|----------------|
| Slow EGO-Swarm | N/A      | 14[25]22     | 0[0]0              | 109.8[113.2]113.5 | 15388.2[15491.5]15486.2 | 11.65[11.67]11.76 | 11.93[11.99]12.97 |
| EGO-Swarm    | N/A      | 64[83]84     | 0.004[0.001]0.01  | 662.5[700.7]787.9 | 90724.9[4611.9]94160.3 | 7.19[7.24]7.28  | 7.38[7.5]7.63    |
| MADER (convex) | N/A      | 15[30]42     | 0.0000.0010.00   | 78.09[74.19]74.74 | 1505.9[1643.6]1683.8 | 6.28[6.25]6.26  | 7.15[7.35]7.04   |
| RMADER (proposed) | 100/130/200 | 0[0]0       | 0.46[0.34]1.751  | 127.7[147.9]190.5 | 2939.4[3712.4]5942.1 | 7.28[7.95]10.35 | 8.4[8.8]11.91    |
|              | (≥100th percentile of δactual and δPC ≥ δmax holds) |          |                    |                |                 |                |
|              | 25[50]/103 | 0[0]0       | 0.001[0.007]0.086 | 69.52[112.0]137.7 | 1844.3[2142.3]3056.2 | 6.80[6.87]7.30 | 7.66[8.02]8.89   |
|              | (≤75th percentile of δactual so δPC ≥ δmax does not hold) |          |                    |                |                 |                |

(a) Collision-free Trajectory Rate  
(b) Travel Time - shaded parts indicate its maximum and minimum value.  
(c) Number of Stops  
(d) Trajectory Smoothness (Acceleration)  
(e) Trajectory Smoothness (Jerk)  
(f) Distribution of δactual in simulations. Due to computer’s computational limits, messages do not travel instantly.

Fig. 6. 100 Flight Simulation Results: Fig. 6a shows RMADER generates collision-free trajectory at 100%, while other state-of-the-art approaches fail when communication delays are introduced. To maintain collision-free trajectory generation, RMADER periodically occupies two trajectories, and other agents need to consider two trajectories as a constraint, which could lead to conservative plans - longer Travel Time and more Avg. Number of Stops. This is a trade-off between safety and performance. MADER reports a few collided trajectory because δactual > 0 ms.

TABLE VI. MADER hardware experiments

| Exp 1 | Exp 2 | Exp 3 | Exp 4 | Exp 5 |
|-------|-------|-------|-------|-------|
| Max vel. [m/s] | 2.7 | 2.5 | 2.7 | 2.7 | 3.0 |
| Avg. travel distance [m] | 70.3 | 67.6 | 61.1 | 61.1 | 65.2 |

Fig. 7. Distribution of δactual in hardware experiments: Both MADER and RMADER were tested in 5 flight experiments. Compared to simulations (see the case δintrod = 0 ms in Fig. 6f), δactual is much larger in hardware.

V. CONCLUSIONS AND FUTURE WORK

We proposed RMADER, a decentralized and asynchronous multiagent trajectory planner that is robust to communication delays. The key property of RMADER is
TABLE VII. RMADER hardware experiments

| Exp. | Max vel. [m/s] | Avg. travel distance [m] |
|------|----------------|-------------------------|
| 6    | 2.6            | 45.6                    |
| 7    | 2.7            | 58.2                    |
| 8    | 2.8            | 58.4                    |
| 9    | 2.7            | 58.3                    |
| 10   | 2.7            | 54.8                    |

$\overline{\text{Max vel. [m/s]}} = \{2.6, 2.7, 2.8, 2.7, 2.7\}$

$\overline{\text{Avg. travel distance [m]}} = \{45.6, 58.2, 58.4, 58.3, 54.8\}$

$\overline{\text{Exp. 6 Exp. 7 Exp. 8 Exp. 9 Exp. 10}}$

$\overline{\text{Max vel. [m/s]}} = \{2.6, 2.7, 2.8, 2.7, 2.7\}$

$\overline{\text{Avg. travel distance [m]}} = \{45.6, 58.2, 58.4, 58.3, 54.8\}$

$\overline{\text{Fig. 8. RMADER Successful Deconfliction under Communication Delays}}$

$\overline{\text{t = 0 s: Agent 1 is following its trajectory}}$

$\overline{\text{t = 0.15 s: Agent 1 and Agent 2 published their trajectory, only 10 ms apart. Due to}}$

$\overline{\text{communication delays each agent did not consider the other trajectory, and thus these}}$

$\overline{\text{two trajectories are in conflicts. Note that we have a 1.5 m-tall boundary box, and}}$

$\overline{\text{thus these trajectories are in collision.}}$

$\overline{\text{t = 1.01 s: During Delay Check both agents detected conflicts and did not commit}}$

$\overline{\text{their trajectory.}}$

$\overline{\text{t = 1.97 s: Collision avoided.}}$

that it guarantees safety even when there are communication delays. RMADER guarantees collision-free trajectories by introducing a delay check mechanism and keeping at least one collision-free trajectory available throughout planning. Simulation and hardware experiments showed RMADER’s robustness to communication delays and the trade-off between safety and performance. Potential future work includes large-scale hardware experiments.

TABLE VIII. Convex MADER vs Nonconvex MADER

| Method        | Computation Time [ms] | $\int |a|^2 dt$ [m²/s²] | $\int |j|^2 dt$ [m³/s³] | Number of Stops | Travel Time [s] | Travel Distance [m] |
|---------------|-----------------------|---------------------|-----------------------|-----------------|----------------|-------------------|
| convex MADER  | 31.08                 | 433.0               | 103.5                 | 2135.0          | 0.18           | 16.05             | 75.24             |
| nonconvex MADER | 99.23               | 724.0               | 441.93                | 20201.8         | 0.16           | 9.93              | 75.80             |

APPENDIX I

CONVEX VS NONCONVEX MADER

Our prior work MADER [4] formulated a nonconvex optimization problem by using both the control points and the separating planes as decision variables [4, Section VI-D]. This could, however, cause expensive onboard computation. Therefore we re-formulated the problem as convex by fixing the separating planes in the optimization (i.e., by not including these planes as decision variables). In addition, to generate smoother trajectories, we added a constraint on the maximum jerk.

We compared both version on a general-purpose-N2 Google Cloud instance with 32 Intel® Core i7. The flight space contains 250 dynamic and static obstacles, and the UAV must fly through the space to reach a goal 75 m away. Maximum velocity/acceleration/jerk are set to $10 \text{ m/s}$, $20 \text{ m/s}^2$, and $30 \text{ m/s}^3$. The performance was measured in terms of Computation Time, trajectory smoothness indicated by $\int |a|^2 dt$ and $\int |j|^2 dt$, Number of Stops, Travel Time, and Travel Distance. The results are shown in Table VIII, where all data is the average of 100 simulations. The notation $\int |a|^2 dt$ and $\int |j|^2 dt$ refers to the time integral of squared norm of the acceleration and jerk along the trajectory, respectively. Higher values therefore represent a less smooth trajectory. Number of Stops is the number of times the UAV had to stop on its way to the goal. Table VIII indicates that convex MADER is computationally less expensive and generates smoother trajectories, but nonconvex MADER performs better in terms of the Number of Stops and Travel Time. Since convex MADER has a computational advantage and can generate smoother trajectories, we implemented convex MADER for MADER and RMADER in all the simulations and hardware experiments in this paper.

ACKNOWLEDGMENT

We would like to thank Nick Rober, Lakshay Sharma, Miguel Calvo-Fullana, Andrea Tagliabue, Dong-Ki Kim, and Jeremy Cai, for their help and insightful comments. This research is funded by Boeing Research & Technology.

REFERENCES

[1] J. Park, J. Kim, I. Jang, and H. J. Kim, “Efficient Multi-Agent Trajectory Planning with Feasibility Guarantee using Relative Bernstein Polynomial,” in 2020 IEEE International Conference on Robotics and Automation (ICRA), May 2020, pp. 434–440, ISSN: 2577-087X.

[2] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” Artificial Intelligence, vol. 219, no. C, pp. 40–66, Feb. 2015. [Online]. Available: https://doi.org/10.1016/j.artint.2014.11.006
[3] D. R. Robinson, R. T. Mar, K. Estabridis, and G. Hewer, “An Efficient Algorithm for Optimal Trajectory Generation for Heterogeneous Multi-Agent Systems in Non-Convex Environments,” IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 1215–1222, Apr. 2018, conference Name: IEEE Robotics and Automation Letters.

[4] J. Tordesillas and J. P. How, “Mader: Trajectory planner in multiagent and dynamic environments,” IEEE Transactions on Robotics, vol. 38, no. 1, pp. 463–476, 2022.

[5] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, “EGO-Swarm: A Fully Autonomous and Decentralized Quadrotor Swarm System in Cluttered Environments,” Nov. 2020, arXiv:2011.04183 [cs] version: 1. [Online]. Available: http://arxiv.org/abs/2011.04183

[6] P. C. Lusk, X. Cai, S. Wadhwania, A. Paris, K. Fathian, and J. P. How, “A Distributed Pipeline for Scalable, Deconflicted Formation Flying,” IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 5213–5220, Oct. 2020, conference Name: IEEE Robotics and Automation Letters.

[7] J. Gielis, A. Shankar, and A. Prorok, “A Critical Review of Communications in Multi-robot Systems,” Current Robotics Reports, Aug. 2022. [Online]. Available: https://doi.org/10.1007/s43154-022-00090-9

[8] Y. Chen, M. Cutler, and J. P. How, “Decoupled multiagent path planning via incremental sequential convex programming,” in 2015 IEEE International Conference on Robotics and Automation (ICRA), May 2015, pp. 5954–5961, iSSN: 1050-4729

[9] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, “Towards Search-based Motion Planning for Micro Aerial Vehicles,” Oct. 2018, arXiv:1810.03071 [cs]. [Online]. Available: http://arxiv.org/abs/1810.03071

[10] J. Park, D. Kim, G. C. Kim, D. Oh, and H. J. Kim, “Online Distributed Trajectory Planning for Quadrotor Swarm With Feasibility Guarantee Using Linear Safe Corridor,” IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 4869–4876, Apr. 2022, conference Name: IEEE Robotics and Automation Letters.

[11] C. Toumieux and A. Lambert, “Decentralized Multi-Agent Planning Using Model Predictive Control and Time-Aware Safe Corridors,” IEEE Robotics and Automation Letters, vol. 7, no. 4, pp. 11110–11117, Oct. 2022, conference Name: IEEE Robotics and Automation Letters.

[12] J. Hou, X. Zhou, Z. Gan, and F. Gao, “Enhanced Decentralized Autonomous Aerial Swarm with Group Planning,” Mar. 2022, arXiv:2203.01069 [cs]. [Online]. Available: http://arxiv.org/abs/2203.01069

[13] M. Cáp, P. Novák, M. Selecký, J. Faigl, and J. Vokřínek, “Asynchronous decentralized prioritized planning for coordination in multi-robot system,” in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nov. 2013, pp. 3822–3829, iSSN: 2153-0866.

[14] S. Baskin and G. Sukhatme, “Asynchronous Real-time Decentralized Multi-Robot Trajectory Planning,” IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2022.

[15] J. Tordesillas and J. P. How, “MINVO Basis: Finding Simplexes with Minimum Volume Enclosing Polynomial Curves,” Computer-Aided Design, vol. 151, no. C, Oct. 2022. [Online]. Available: https://doi.org/10.1016/j.cad.2022.103341