Dynamic DEC LOS RRT

Saahil Parikh, Yi Wang

***Abstract*— The problem of ensuring the safety of multi-agent motion when agents cannot contact each other is essential for path planning in the real world. It is difficult to control the agents experiencing instantaneous stops to avoid collisions. To address this problem, we develop a new algorithm Dynamic DEC-LOS-RRT based on the existing approach DEC-LOS-RRT. A new radial velocity set is defined around the corners of each obstacle to make robots stop smoothly. This is implemented by constraining the configuration space of each robot and adding constraints to the local planning algorithm. Simulation and hardware show our algorithm guarantees the effectiveness of path planning and safety for decentralized planning for multi-agent systems.**

I. INTRODUCTION

In various fields, autonomous agents can be operated in restricted, highly structured environments, e.g. warehouses[1], urban streets[2] and other roads[3]. Efficient and safe path planning for autonomous mobile vehicles is fundamental to achieve a multitude of tasks in a large-scale team. This problem, aiming at getting collision-free paths to target position, exists with three following challenges: communication constraints, fleets in large scale and complex environment. It is obvious that a decentralized path planning algorithms must be able to address these challenges, i.e., have the ability to safely reach the goal in a constrained environment. In addition, the vehicle dynamics constraints should also be considered in order to allow realistic application of the algorithm.

In order to navigate to the target, the individual agents must meet the interaction constraints between the robots, the fixed obstacle constraints in the surrounding environment, and the vehicle dynamics constraints. Such a complex environment leads not only to increased complexity in path planning with obstacle constraints but also to communication limitations between different agents are not in line-of-sight(LOS) about interaction constraints. This limitation is typical in reality, as the position signal of one robot cannot pass through the object to be detected by the rest of the robots for real-time, such as radar and optical cameras. In the past work, agents are designed to avoid collisions through instantaneous stop or other speed settings that are difficult to achieve in practical system.

In this paper, a decentralized path planning method satisfying dynamical constraints in complex environments is proposed for multi-agent. We propose a simplified, implementable dynamics model where agents can perform continuous acceleration and deceleration processes to avoid collisions, even not in LOS. The setting is illustrated in Fig 1. To satisfy the agent's dynamics constraints and allow the agent to execute the path planning algorithm in real-world, we force the agent to perform tasks at a specific speed setting. The speed of the agent is determined by the distance of the agent from the corner point of the obstacle. Under such dynamical constraints, we develop a decentralized path planning algorithm for multi-agent requiring to reach the desired target location in a determined physical workspace.

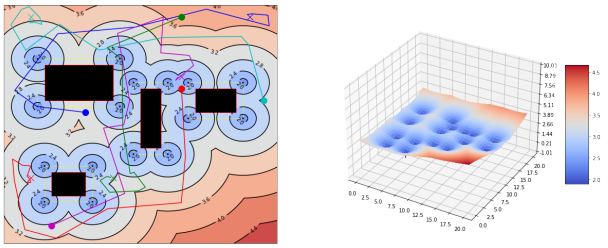

Fig. 1: Left: An example of the environment and the newly defined speed set. Right: Velocity manifold heat map The agent is asked to plan a path without crashing and reach the target point. There are δ-obstacles in the environment and some new set of velocities are introduced as dynamic constraints. The velocity set is set as a series of concentric circles centered on the corner points of the obstacle. The smaller the radius of the velocity circle, the lower the velocity at which the agent arrives.

II. RELATED WORK

The application of automated mobile robots has recently grown substantially in various fields and has been researched extensively [4]. Classical multi-agent path planning algorithms can be categorized broadly as centralized or decentralized. The centralized approach is proposed based on a planning center that can detect the current state of all robots with respect to the desired state and return a path for all robots. [5] develops a complete method for solving a general class of robots considering dynamic searching in space of priority for the robots and use A^* to generate the path plan. The decentralized approach provides attractive improvements over centralized approaches, resulting from its ability to reduce the computational burden and relieve the dependence on the central unit. In many decentralized algorithms, agents consider only their neighbors in the same communication network [6].

When robots can pass information to each other, there exist many methods for navigating with free-collisions. [7] proposed a strategy of making the robots in the team check the path at each step to avoid collisions, which is workable but causes a large computational cost. Another noteworthy approach is suggested by V. R. Desaraju [8], in which a

merit-based token is applied to the dynamic update of the agent state, letting the agent retain the ability of fast path planning in complex environments. These algorithms are executed under the condition that the agents are in LOS. When agents are located on either side of an obstacle corner, the Euclidean distance between them is small but they are invisible, which may cause collisions. [9] developed an algorithm with expansion of boundary constraint on obstacles that allows agents to avoid collisions and still reach the target location when they are not in LOS.

These methods, however, are typically ignoring the dynamical constraints of the agent. Although the agent detects the remaining agents outside the distance threshold and stops its action to avoid a collision, the assumption that the velocity can change abruptly is not valid in reality because the acceleration of the agent's motion is upper bounded. Our approach, through a new radial velocity set to add the constraints in local planning algorithm, guarantee that the agent can safely reach the target under a smooth speed change.

III. PROBLEM STATEMENT AND DEFINITIONS

In this section, we define some relevant variables, terms, and our problem mathematically.

A. Notation definition

We define the agents motion and communication state below.

a) Robot path model: For an specific agent i that agent is in a sub-graph with all other agents j in LOS. Therefore, obstacles in the environment can cut off the communication between agents.

b) Safety Constraint: We will define a safety constraint to ensure that robots are always at least a distance ϵ away from one another. This can be seen as $\min_{robots} pairwise > \epsilon$.

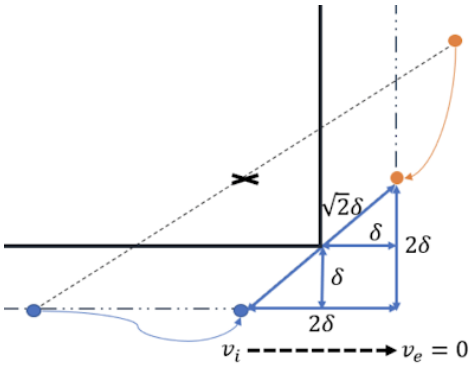

Fig. 2: The minimum distance from point of collision two agents can ever be. This is the worst case scenario.

B. Motivation

Our objective is to figure out how fast a robot can be moving in an environment to ensure that it doesn't collide with other dynamic agents. Using the motivation in Fig [2] we can start to see that the worst case scenario between two agents is that the they will collide a distance of 2δ from when they first saw each other and where δ is defined by the factor by which we expand our obstacles to make delta obstacles [9].

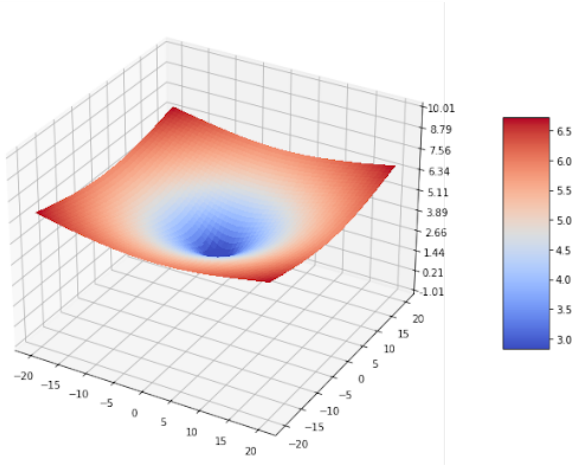

Fig. 3: Single well with the formula $V_f^2 = V_0^2 + 2a_{max}\Delta x$

C. Maximum Globally safe velocity

Using the kinematic equation $V_f^2 = V_0^2 + 2a_{max}\Delta x$ we can start to derive a maximum constraint on the velocity of an agent. Given our worst case scenario this magnitude defines how fast an agent can go to ensure that it will never hit another agent with velocity.

$$\begin{aligned} V_f^2 &= V_0^2 + 2a_{max}\Delta x \\ 0 &= V_0^2 + 2a_{max}\Delta x \\ V_0 &= \sqrt{2a_{max}\Delta x} \\ &= \sqrt{4a_{max}\delta} \\ &= 2\sqrt{a_{max}\delta} \end{aligned}$$

Using a worst case of 2δ we can define our velocity constraint map as $V = \max(2\sqrt{a_{max}\delta}, \sqrt{2a_{max}\Delta x})$ shown in figure 3 Furthermore, using the ϵ safety constraint we can reform the solution into.

$$\begin{aligned} V_0 &= \sqrt{2a_{max}\Delta x} \\ &= \sqrt{2a_{max}(2\delta - \epsilon/2)} \\ &= \sqrt{4a_{max}\delta - a_{max}\epsilon} \end{aligned}$$

proving that robots will always be at least $\sqrt{2}\epsilon/2$ away from each other. For succinctness, the rest of the paper will use the equation derived without the ϵ constraint.

D. Creating a manifold on the configuration space

Now that we have defined the velocity that an agent can go at given a this constraint, we can start to define how the environment should be constrained if there were multiple of these constraint as $\min_{edges}(\max(2\sqrt{a_{max}\delta}, \sqrt{2a_{max}\Delta x}))$. This is seen in figure 4.

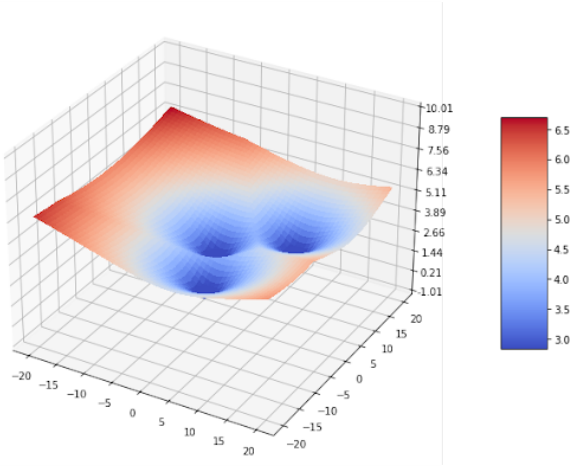

Fig. 4: Multiple wells with the formula $\min_{edges}(\max(2\sqrt{a_{max}\delta}, \sqrt{2a_{max}\Delta x}))$

E. Global map

Notice that the manifold is a function of $\sqrt{a_{max}}$. Therefore, in practice we can define one mapping function given the edges in the environment and then multiply $\sqrt{a_{max}}$ to it for each agent.

$$map = \min_{edges}(\max(2\sqrt{\delta}, \sqrt{2\Delta x}))$$
$$manifold = \min_{edges}(\max(2\sqrt{a_{max}\delta}, \sqrt{2a_{max}\Delta x}))$$

IV. RESULTS

In this section, we present and evaluate our method on a foreknown workspace with static rectangle obstacles and N agents. We also run our algorithm in real world to prove the feasibility in practical environments.

A. Simulation Setup

a) Implementation details:: The implementation for our approach, Dynamic DEC LOS RRT, were achieved in Python.

b) Workspace and simulation parameters:: The workspace with scales 20×20 can be shown in Fig5. Agents has initial and goal positions given randomly outside of the δ-obstacles. In Fig5 the initial and goal positions for 7 agents are [index of agent:(x,y) in initial position-(x,y) in goal position]: 1: (5.5, 18.)-(15., 15.), 2: (3., 5.)-(10., 2.), 3: (8., 11.)-(15., 18.), 4: (15., 2.)-(3., 16.), 5: (10., 17.)-(19., 6.), 6: (4., 16.)-(14., 10.), 7:(15., 18.)-(10., 15.). Other setting and results are shown in Appendix. The coordinates of the diagonal points of the obstacle are [index of obstacles:(x,y) for lower right corner-(x,y) for upper right corner]: 1: (3, 7)-(5, 10), 2: (6, 2)- (8, 4), 3:(12, 4)-(18, 8), 4: (6, 13)-(8, 19), 5: (12, 13)-(14, 19), 6: (16, 15)-(19, 17). $\delta_m in$ for 3 and 5 agents is set as 0.7; $\delta_m in$ for 3 and 5 agents is set as 0.45. The initial iteration k is set as 5000. When agent arrives the goal position, it will keep stillness.

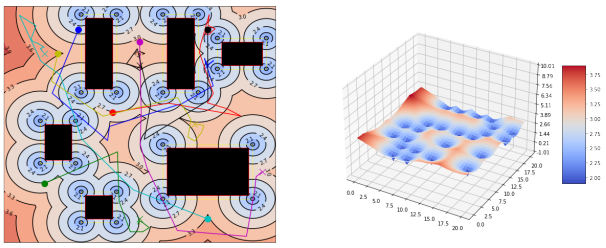

Fig. 5: 7 agents navigating to their target positions by Dynamic DEC LOS RRT with their velocity manifolds. The filled boxes represent obstacles, and the dashed boxes represent δ-obstacles. • represents the original positions for agents; $\times$ represents the goal positions for agents.

B. Main Simulation Results

For N = 3, 5, 7, 10, the agents, by executing our algorithm, successfully and safely reach the target under environmental, dynamical and interaction constraints. The right figure show the manifold of agent velocity. Such an environment is considered safer when the agent is far from the corner point of the barrier, where the agent moves at a relatively large speed. As the agent approaches the corner point, i.e., reducing the distance from the remaining invisible agents, the agent's velocity gradually decreases, which enables the agent to stop before colliding. Furthermore, the agent performs a deceleration-stop process with a smooth velocity profile before possible collisions.

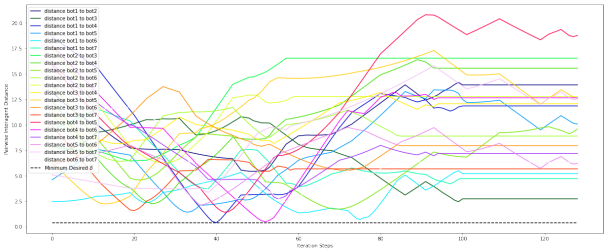

Fig. 6: Inter-agent distances with no violations for 7 agents. All pairwise distance is greater than $\delta_m in$.

Fig 6 shows the distance between each pair of agents at the same time for a run with 7 agents. $\delta_m in$ is the distance threshold at which the agents are visible and cross each other. There is no point at which the distance between pairs of agents falls below the minimum threshold of $\delta_m in$. Agents executing our method are able to obtain enough buffer distance to adjust their own speed even if the rest of the agents are not visible temporarily.

As the number of robots increases, space becomes more crowded which leads that $\delta_m in$ reduced from 0.7 to 0.5. As the number of obstacles increases with the number of agents, the difficulty for agents to obtain collision-free paths increases, and the search time also increases shown in Fig7. As shown in Fig8, the consumption of average iteration for N agents from the starting point to the end point changes not significantly, for the fact that the paths are all randomly generated.

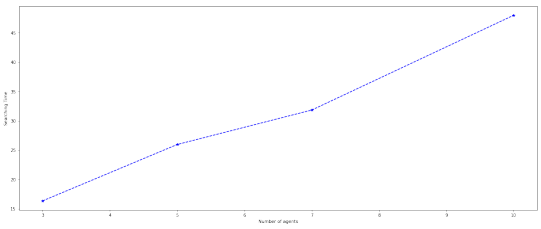

Fig. 7: The relationship between search time and number of agents.

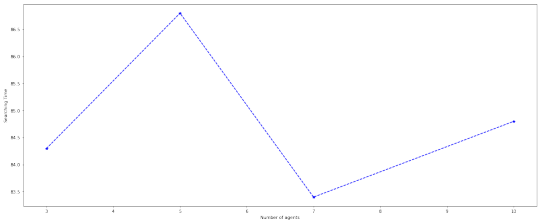

Fig. 8: The relationship between average iteration for N agents and number of agents.

C. Hardware Setup

a) Vehicle model: The following unicycle model is used:

$$\dot{x} = V cos\phi \tag{1}$$

$$\dot{y} = V sin\phi \tag{2}$$

$$\dot{\phi} = \omega \tag{3}$$

where $\dot{x}$ represents the velocity along with x-axis; $\dot{y}$ represents the velocity along with y-axis and $\dot{\phi}$ represents the angular velocity.

The input of controller is set as:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4}$$

where u_1 is the linear speed and u_2 is rotational speed. To adapt to practical needs, the input in Eq4 is transformed as dynamics below:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r/2 & r/2 \\ r/l & -r/l \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \tag{5}$$

where ω_r and ω_l replacing v and ω represents the angular velocities for the right and left wheel.

b) Workspace and hardware parameters: The dimensions of robots are 11 cm wide, 10 cm long, and 7cm tall. The maximum v is 20cm/s and the maximum ω is 3.6 rad/s. The obstacles information can be accessible to agents in advance, so we set obstacles as projection.

c) Simulation with real parameters: For simulation with unicycle model, we use a remote platform named Robotarium. The parameters are set in the same way as in real hardware shown before.

D. Main Hardware Results

The simulation for unicycle model executes before the algorithm applied in hardware. Fig9 shows the results of the simulation; Fig10 shows the results of path planning for 7 robots in a real environment. All agents run almost the same trajectory in reality as in the simulation and reach the end without collision, which proves the reliability of our algorithm.

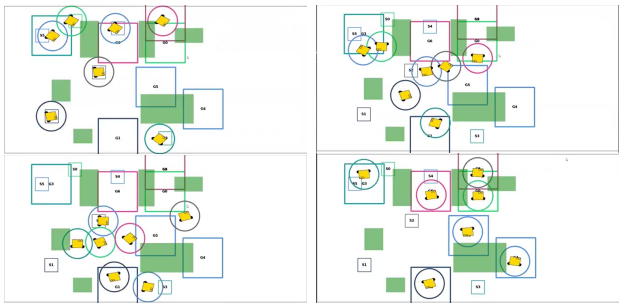

Fig. 9: 7 agents running in simulation for unicycle model. They arrive their goals without collisions.

However, some jams occur during the agent's change of direction. Since the Dynamic DEC LOS RRT runs on a 2-dimensional workspace, i.e., the position of the agents, to plan the path, it is difficult to get a smooth path. The algorithm is attempted to run on a 4-dimensional workspace, i.e., the position and velocity of the agents, but the correct path was difficult to find due to large numbers of sampling points.

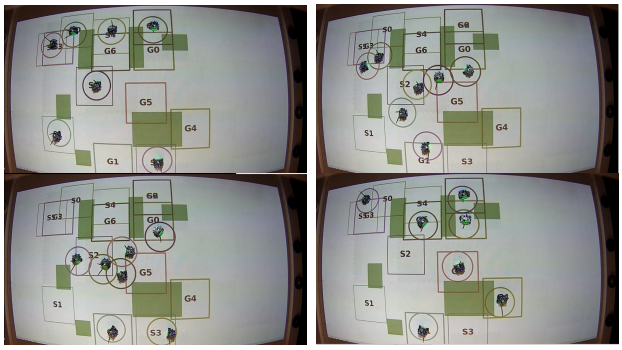

Fig. 10: 7 agents running in real world. They arrive their goals without collisions.

E. Conclusion

The simulation and hardware results present our algorithm performs indeed ensure that the agents can safely and effectively navigate to their goal positions with smooth velocity at each time step. Compared to the DEC-LOS-RRT, we address the assumption of instantaneous stops and set a more efficient set of speeds, allowing the implementation on actual ground robots. As the number of robots grows, searching time increases in acceptable range and the average iteration does not change obviously. In the future, we will work on

expanding the dimensionality of the workspace and improve the efficiency of the algorithm.

V. DISCUSSION

A. Summary

In this paper, we have develop a effective and safe way for distributed multi-agent path planning to satisfy environment, dynamic and interaction constraints. Our method builds a velocity set as dynamic constraint around obstacle corner points upon DEC-LOS-RRT. The performance on simulation and hardware displays that agents running our algorithm can successfully plan path from initial position to goal position. Although we found that there is an imperfection in the velocity direction and the paths generated from the planning are not smooth enough, our algorithm still is realistic in its implementation and effectively achieves the vehicle dynamics constraint.

B. Difficulties and Future Work

The limitation of the lab space forced us to use the remote platform Robotarium for our experiments. This prevented us from getting immediate feedback and making accurate adjustments to the parameters based on the actual hardware. Fortunately, the platform provides a very stable controller, making the simulation results nearly identical to those presented in hardware.

To obtain smooth paths, we tried to implement our algorithm in 4-dimensions workspace, but the agent was not able to find the correct path within the iterations we set currently. We will improve on this in the future likes defining constraints over the configuration space to help agents reach the goal. eg. Velocity Bowl around goal. In addition, we want to use the gradient of the constraint map instead of just the magnitude in the future. Solving for the set of vectors v_0 given a, v_f, and would be a step closer to that goal. This paper uses the magnitude of velocity in it's derivations.

ACKNOWLEDGMENT

This work was supported by a remote robot platform Robotarium. We would like to thank the owner for providing the hardware.

REFERENCES

[1] "What's Going on With Amazon's "High-Tech" Warehouse Robots?", https://spectrum.ieee.org/whats-going-on-with-amazons-hightech-warehouse-robots, accessed: 2022-5-12

[2] "Urban Autonomy: The Next Stage Of Self-Driving", https://www.forbes.com/sites/forbestechcouncil/2020/08/14/urban-autonomy-the-next-stage-of-self-driving/?sh=3217da067628, accessed:2022-5-12

[3] "Exploding growth: Autonomous delivery market set to take off", https://www.freightwaves.com/news/exploding-growth-autonomous-delivery-market-set-to-take-off, accessed:2022-5-12

[4] M. Raibail et al., "Decentralized Multi-Robot Collision Avoidance: A Systematic Review from 2015 to 2021," Symmetry, vol. 14, no. 3, p. 610, Mar. 2022

[5] R. Luna and K. E. Bekris, "Efficient and complete centralized multi-robot path planning," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 3268-3275

[6] W. Wu, S. Bhattacharya and A. Prorok, "Multi-Robot Path Deconfliction through Prioritization by Path Prospects," 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 9809-9815

[7] P. Scerri, S. Owens, B. Yu, and K. Sycara, "A decentralized approach to space deconfliction," in Proc. 10th Int Information Fusion Conf, 2007, pp. 1–8

[8] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees," 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 4956-4961

[9] V. Tuck, Y. V. Pant, S. A. Seshia and S. S. Sastry, "DEC-LOS-RRT: Decentralized Path Planning for Multi-robot Systems with Line-of-sight Constrained Communication," 2021 IEEE Conference on Control Technology and Applications (CCTA), 2021, pp. 103-110

APPENDIX

Here we will show the result of 3, 5 and 10 agents. The path and velocity manifold are shown in Fig11 and the Inter-agent distances are shown in Fig12. The simulation and hardware results are presented in our website.

a) 3 agents: The initial and goal positions for 3 agents are [index of agent:(x,y) in initial position-(x,y) in goal position]: 1: (1., 1.)-(19., 19.), 2:(18., 18.)-(2., 2.), 3: (15., 5.)-(5., 14.5). The coordinates of the diagonal points of the obstacle are [index of obstacles:(x,y) for lower right corner-(x,y) for upper right corner]: 1: (3, 3)-(10, 10), 2: (16, 16)-(17, 17), 3: (10, 15)-(12, 17).

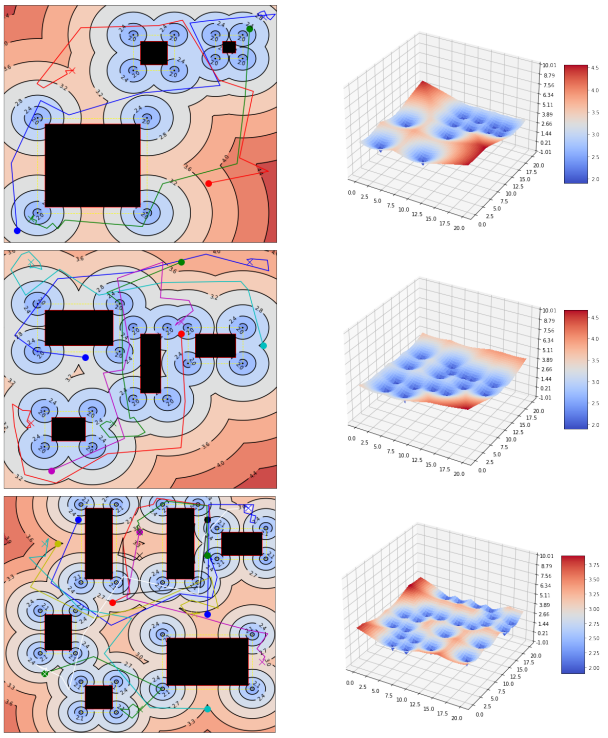

Fig. 11: 3, 5 and 10 agents running in real world. They arrive their goals without collisions.

b) 5 agents: The initial and goal positions for 5 agents are [index of agent:(x,y) in initial position-(x,y) in goal position]: 1: (6., 11.)-(18., 19.), 2:(13., 19.)-(8., 5.5), 3: (13., 13.)-(2., 5.5), 4:(19., 12.)-(2., 19.), 5: (3.5, 1.5)-(13., 14.). The coordinates of the diagonal points of the obstacle are [index of obstacles:(x,y) for lower right corner-(x,y) for upper right corner]: 1: (3., 12.)-(8., 15.), 2: (3.5, 4.)-(6., 6.), 3: (10., 8.)-(11.5, 13.), 4: (14., 11.)-(17., 13.).

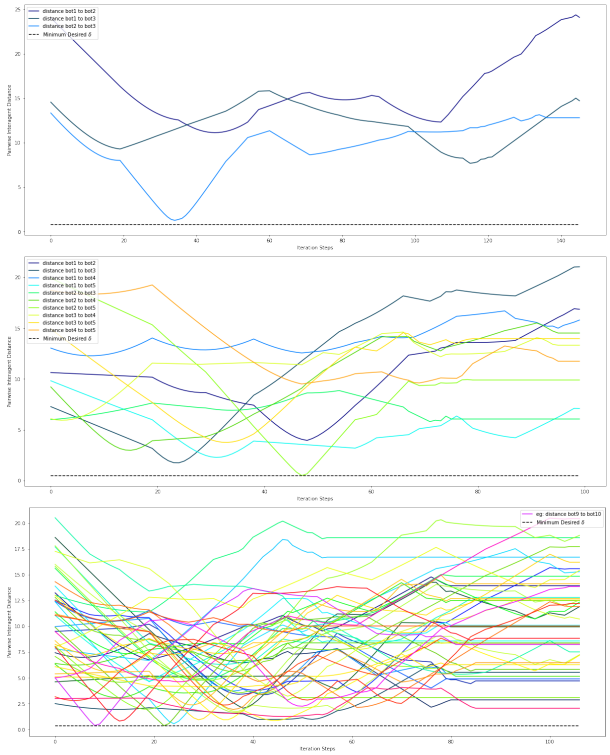

Fig. 12: Inter-agent distances with no violations for 3, 5 and 10 agents. All pairwise distance is greater than $\delta_m in$.

c) 10 agents: The initial and goal positions for 10 agents are [index of agent:(x,y) in initial position-(x,y) in goal position]: 1: (5.5, 18.)-(15., 15.), 2: (3., 5.)-(10., 2.), 3: (8., 11.)-(15., 18.), 4: (15., 2.)-(3., 16.), 5: (10., 17.)-(19., 6.), 6: (4., 16.)-(14., 10.), 7:(15., 18.)-(10., 15.), 8: (18., 19.)-(3., 5.), 9:(15., 10.)-(18., 19.), 10: (15., 15.)-(10., 17.). The coordinates of the diagonal points of the obstacle are [index of obstacles:(x,y) for lower right corner-(x,y) for upper right corner]: 1: (3, 7)-(5, 10), 2: (6, 2)- (8, 4), 3:(12, 4)-(18, 8), 4: (6, 13)-(8, 19), 5: (12, 13)-(14, 19), 6: (16, 15)-(19, 17).

d) Codes: https://github.com/SaahilParikh/DYNAMIC-DEC-LOS-RRT

e) Website: https://saahilparikh.github.io/DYNAMIC-DEC-LOS-RRT/