Multi-Agent Collision Avoidance Method Using Fuzzy Risk Estimation and Information Sharing in Unknown Environments

Yigit Can Dundar
Dept. of Computer Science
UiT - The Arctic University of Norway
Tromsø, Norway
yigit.c.dundar@uit.no

Bernt Arild Bremdal
Dept. of Computer Science and Computational Engineering UiT - The Arctic University of Norway
Narvik, Norway
bernt.a.bremdal@uit.no

Anne Hakansson
Dept. of Computer Science
UiT - The Arctic University of Norway
Tromsø, Norway
anne.hakansson@uit.no

Randi Karlsen
Dept. of Computer Science
UiT - The Arctic University of Norway
Tromsø, Norway
randi.karlsen@uit.no

***Abstract*—Automated vehicles within Industry 4.0 are used as logistics units where they can move resources from one place to another safely and efficiently. T he a utomated v ehicles can be tasked to work in unknown environments where collision-free navigation is challenging due to uncertainty and lack of environmental information. Collisions can damage equipment and may even cause harm to human workers sharing the same space. In order to carry out tasks and avoid collisions in unknown environments, automated vehicles need means of self regulation and environmental awareness through sensors. This paper presents a multi-agent collision avoidance method using fuzzy risk estimation and information sharing for automated vehicles navigating unknown environments. The automated vehicles exchange information about their current state and GPS location with other automated vehicles and use fuzzy collision risk estimation based on sensor data to avoid collisions with static obstacles and each other. Additionally, the automated vehicles collectively learn about their work environment over time by creating a virtual map of the environment through shared information. To test and evaluate the method, a series of simulated tests are carried out. The results from tests show that the automated vehicles are able to avoid collisions while also accurately mapping an initially unknown work environment using the method.**

***Index Terms*—collision avoidance, fuzzy risk estimation, information sharing, multi-agent systems, unknown environment**

I. Introduction

Industry 4.0 describes the evolution of computer-controlled automated facilities into cyber-physical systems that gather and analyze data for intelligent autonomous decision making [1]. Within Industry 4.0, a common warehouse logistics solution is the usage of Automated Vehicles (AVs). AVs have proven adequate to handle a large spectrum of tasks within the material flow area [2]. AVs can move resources like production materials, tools and parts using safe and efficient path finding and collision avoidance solutions. Avoiding collisions in the environment can save maintenance time and resources, while also creating a safe work environment for other AVs and human beings.

AVs that are tasked to work in unknown environments are faced with navigational uncertainty due to lack of environmental information. Since finding a collision free path requires environmental information, planning a path in an unknown environment before run-time is challenging and can be inaccurate. Additionally, the multi-agent aspect of the environment can lead to further collision avoidance complexity where other vehicles navigating the same environment act as dynamic and unpredictable obstacles. To compensate for the lack of information, the AVs need to rely on their sensors and each other, rather than pre-processed paths, to avoid collisions and navigate an unknown work environment. Collisions, if not avoided, can damage equipment, cost time and may even harm human workers sharing the environment with the AVs.

This paper presents a multi-agent collision avoidance method for use in unknown environments based on fuzzy risk estimation and information sharing. The AVs are managed as a Multi-Agent System (MAS), where agents coordinate with each other to proactively carry out both individual and collective goals [3]. Using the method, the AVs can avoid obstacles through a fuzzy risk estimation of a collision. The risk estimation is based on fuzzy logic, where a rule base is used that maps distance to an obstacle and current vehicle velocity into three distinct risk levels. Based on the risk level, the vehicles adjust their speed and make changes to their trajectory to avoid obstacles. Additionally, the AVs exchange

GPS location and status information among themselves to avoid collisions with each other. The vehicles also record the estimated locations of detected obstacles to create a virtual map of the initially unknown environment. The method is evaluated through three experimental test cases. The findings are compared to related collision avoidance approaches to highlight differences, similarities, strengths and shortcomings of the collision avoidance method.

The contribution of this research paper is the multi-agent collision avoidance method based on fuzzy risk estimation and information sharing. The goal of the method is to broaden the possible work environments that the AVs can be deployed in. The method aims to achieve this goal by providing means of autonomous navigation and collision avoidance to AVs so that they are not entirely reliant on environmental information. The method can improve navigational flexibility, autonomy and safety of AVs used in and out of Industry 4.0.

II. Related Work

The collision avoidance problem has seen various types of solutions in various types of agent and environmental configurations.

In mapless and unknown environments, a common solution for avoiding congestions and finding paths is to use reinforcement learning. A reinforcement learning based mapless motion planner by Marchesini et al. [4] shows that a single agent can be virtually trained to find collision free paths in a mapless environment. Similarly, a discrete deep-reinforcement learning method by Tai et al. [5] also trains a single agent in a virtual mapless environment to find paths without collisions. Another solution to avoiding collisions in unknown environments is to use fuzzy logic. A single agent solution by Beom et al. [6] utilizes fuzzy logic to determine the distant safety levels of an obstacle to a vehicle for collision-free navigation. Contrary to related work avoiding collisions in unknown and mapless environments, the method presented in this paper provides multi-agent collision avoidance rather than focusing on a single agent.

In multi-agent environments, several solutions have been explored in the past utilizing fuzzy logic and information sharing for multi-agent path finding and collision avoidance purposes. Fuzzy logic has been used to determine the relative position of obstacles for multi-agent path finding by Kumar et al. [7], to find stable and collision free paths by Bogdan et al. [8], and to determine critical cases for collision avoidance by Souliman et al. [9]. A different approach by Lin et al. [10] utilizes fuzzy logic to detect and escape distinct local minimum trap cases in a complex and known environment. Additionally, information sharing among agents was used to resolve intersection and collision conflicts by Makarem et al. [11], and to efficiently navigate an environment while avoiding collisions by Subramanian et al. [12]. Information sharing has also been utilized for creating a virtual map of an initially unknown environment by Wang et al. [13]. Overall, multi-agent path finding and information sharing based solutions assume that the environment is known and fully observable for navigation and collision avoidance purposes, except for the virtual map solution [13]. In contrast to the multi-agent path finding and information sharing related work, the method presented in this paper assumes an initially unknown work environment for multiple agents for collision avoidance. Similar to the virtual map solution [13], the method also creates a virtual map of the initially unknown environment through information sharing.

The collision avoidance method, presented in this paper, builds upon the dynamic path finding method [14]. The dynamic path finding method uses attractive and repulsive forces calculated during run-time to avoid collisions and move towards a target destination. The target destination attracts the agent to move towards it (attractive force) and the obstacles in the environment repel the agent away from them (repelling force). The collision avoidance method uses the dynamic path finding algorithm for navigation and utilizes fuzzy logic and information sharing on top of it to improve collision avoidance for multiple agents in unknown work environments.

III. The Multi-Agent Collision Avoidance Method

The multi-agent collision avoidance method is a step-wise method that uses fuzzy risk estimation coupled with multi-agent information sharing to avoid collisions. The fuzzy risk estimation uses velocity and obstacle distance measurements to determine the risk of collision to the obstacles. The risk estimation is then used by the AVs to adjust their speed and their current trajectory to avoid collisions and escape local minimum traps. The method also allows the agents to exchange their location and status information to identify and avoid collisions with other agents. The agents also share the locations of detected obstacles with a software agent to create a virtual map of the environment. To carry out the tasks mentioned above, the method includes the following four steps:

- Obstacle detection: Detect and measure distances to obstacles in range
- Path Calculation & Fuzzy Risk Estimation: Calculate a collision free path based on fuzzy risk estimation
- Path Correction: Make adjustments to the path when local minimum traps are encountered
- Information Sharing & Identification: Share location and status information with other agents and identify other agents

The following sub-sections will provide in-depth information into each step respectively.

A. Obstacle detection

In order to avoid a collision with an obstacle, the AVs need to first detect the obstacle using sensors. The AVs are assumed to be equipped with sensors capable of detecting obstacles and measuring the direction and the distance to the detected obstacles.

The AVs gather sensor data which includes a direction vector towards an obstacle and the closest distance to it every 0.2 seconds. The data is then used to find a safe path

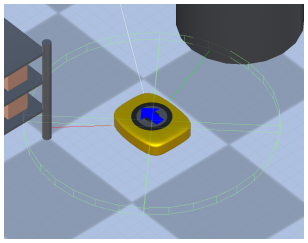

Fig. 1. Automated vehicle sensor setup with range

away from the obstacles within range. As seen in Fig. 1, the simulated AVs for testing the method are equipped with 4 LiDAR sensors located to the front, left side, right side and back of the vehicle. The range of the sensors are highlighted with the green circle around the vehicle (Fig. 1) and the radius of the sensors are set to be twice the front-to-back length of the vehicle.

B. Path Calculation & Fuzzy Risk Estimation

Once the obstacle direction and distance data is gathered from the sensors, a collision-free path is calculated. The attractive and repulsive forces are calculated at each sensor scan interval (every 0.2 seconds). The attractive force is the direction from the AV towards the target destination. The direction is stored as a normalized coordinate vector with x, y and z coordinates. When added to the current GPS location vector of the AV, the resulting vector is used as an arbitrary target destination for the AV to move towards. The repulsive force is the direction away from obstacles relative to the current GPS location of the AV. The gathered obstacle direction data is inverted and then normalized to a coordinate vector to generate the repulsive force away from an obstacle within detection range. When the repulsive force is added to the current GPS location vector of the AV, the resulting vector provides an arbitrary path to the AV that the agent follows to move away from the obstacle.

The collision avoidance method estimates the risk of collision with obstacles based on the distance to the obstacle and current directional velocity of the vehicle. The distance and velocity input pair is fuzzified and mapped to a fuzzy rule base as seen in Table I.

TABLE I.
FUZZY LOGIC RULE-BASE FOR DISTANCE AND VELOCITY INPUT PAIRS

Velocity/Distance	**Close**	**Medium**	**Far**
Slow	Caution	Safe	Safe
Moderate	Risky	Caution	Safe
Fast	Risky	Risky	Caution

There are nine total combinations for distance and velocity input pairs and three estimation types. The estimation types are: Safe, Caution and Risky. For a Safe estimation, the vehicle tries to maximize its current speed while slightly adjusting its trajectory. For Caution estimation, the vehicle moderately slows down and also moderately adjusts its current trajectory.

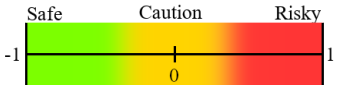

Fig. 2. Final fuzzy risk estimation

Risky estimation forces the vehicle to drastically slow down and change trajectory in order to avoid a collision with the obstacle.

1) Fuzzification process: In order to map the sensory inputs to the rule base, the inputs first need to be normalized to a range between 0 to 1. For both inputs, distance and velocity, (1) and (2) are used respectively.

$$D_n = \frac{D_{cur} - D_{min}}{D_{max} - D_{min}} \tag{1}$$

$$V_n = \frac{V_{cur} - V_{min}}{V_{max} - V_{min}} \tag{2}$$

Both D_n and V_n represent the normalized values which are based on the current distance (D_{cur}) and velocity (D_{cur}) values' relation to the minimum (D_{min}, V_{min}) and maximum distance and velocity (D_{max}, V_{max}) that can be measured by the sensors. The value ranges that represent distance classifications can be seen in (3) and value ranges that represent velocity classifications can be seen in (4).

$$\begin{aligned} &\text{Close:} && 0 \leq D_n \leq 0.33 \\ &\text{Medium:} && 0.33 < D_n \leq 0.66 \\ &\text{Far:} && 0.66 < D_n \leq 1 \end{aligned} \tag{3}$$

$$\begin{aligned} &\text{Slow:} && 0 \leq V_n \leq 0.33 \\ &\text{Moderate:} && 0.33 < V_n \leq 0.66 \\ &\text{Fast:} && 0.66 < V_n \leq 1 \end{aligned} \tag{4}$$

After classifying distance and velocity inputs, computing the risk estimation is done via subtracting the normalized distance value from the normalized velocity value as seen in (5).

$$R = V_n - D_n \tag{5}$$

The final risk value R is a floating number value between -1 and 1. The spectrum seen in Fig. 2 represents the final risk value range with the colors signifying "Risky" estimation for red areas (value range from 1 to 0.33), "Caution" estimation for yellow areas (value range from 0.33 to -0.33) and "Safe" estimation for green areas (value range from -0.33 to -1). The vehicle uses the risk estimation to adjust its current velocity target. If there are multiple obstacles in vicinity, the calculated risk estimation with the higher risk value is used for adjusting the velocity. For example, if there are three obstacles in range with "Risky", "Safe", "Caution" risk estimations respectively, then the "Risky" estimation will be the basis of velocity adjustment.

The risk estimation process is calculated for each obstacle within detection range. As part of this process, each repulsive force used in the Dynamic Path Finding method (that pushes the agent away from itself), is altered based on the risk

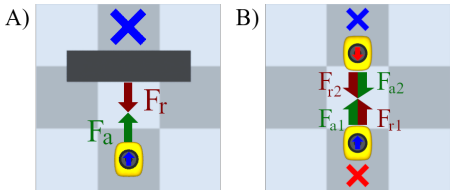

Fig. 3. Local minimum example

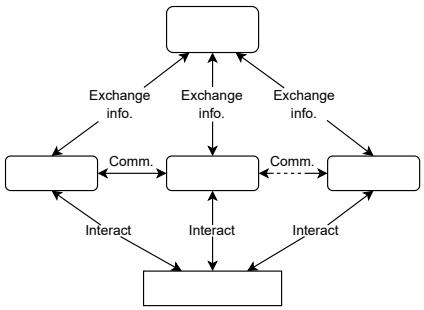

Fig. 4. Multi-agent architecture and hierarchy

estimation. When the risk estimation for a specific obstacle is "Risky", the repulsive force coming from that obstacle will be at maximum 100% effectiveness in magnitude. Alternatively, the magnitude of repulsive forces with a "Caution" estimation will be at maximum 50% and "Safe" estimations will be at maximum 10% effectiveness. The alteration of repulsive forces helps the vehicle adjust its current trajectory to be safer in riskier collision avoidance situations and more efficient in safer collision avoidance situations.

C. Path Correction

A local minimum trap can be encountered when the attractive and repulsive forces overlap or negate each other. Force overlap can cause the agent to move directly towards an obstacle if the attractive force is larger than the repulsive force, which can result in a collision. Force negation can cause the agent to get stuck in one place creating a deadlock situation. Additionally, in a multi-agent environment, local minimum traps can occur in a dynamic and unpredictable manner due to the presence of other agents in the environment.

Fig. 3 shows two examples of common local minimum traps. In Fig. 3 A, the agent is trying to move to the northern side of the grid based on the attractive force denoted as F_a and is blocked by an obstacle which is generating a repulsive force denoted as F_r. The two forces, F_a and F_r, negate each other, causing the agent to remain in place. In Fig. 3 B, two agents are moving head-to-head towards the polar opposite side of the grid where they cause a local minimum trap for each other. The forces F_{a1} and F_{r2} negate each other, and the forces F_{a2} and F_{r1} negate each other as well. Thus causing a deadlock.

To avoid local minimum traps, the method provides a heading correction solution which also uses the distance fuzzy classification. Based on the current distance classification to the obstacle, that is causing the trap, the heading of the vehicle is skewed to either the right or the left side with the skew angle ranging from $35°$ to $45°$ based on the distance. The riskier the distance to the obstacle the larger the angle will become. The correction is always made to the opposite side of the obstacle relative to the vehicle. For instance: if an obstacle causing a trap is to the left of the vehicle, the heading will be skewed to the right of the vehicle. The heading correction allows the agent to escape many dynamic and static local minimum traps that can be encountered in an unknown, multi-agent working environment.

D. Information Sharing & Identification

In the last step of the method, the AVs exchange information about their current state, location and detected obstacle locations. During this step, the agents identify each other to aid collision avoidance and generate a virtual map of the unknown environment. Within the multi-agent system architecture each AV is represented as an agent. There are two different agent types which communicate with each other with distinct objectives:

- Worker agent (AVs): Tasked with navigating to requested destinations in the environment while also estimating and gathering obstacle location information
- Manager agent (Software): Tasked with distributing target destinations to worker agents and gathering information from them to create a virtual map of the environment

While the worker agent (the AV) can interact with the environment directly using its sensors and actuators (wheel motors), the manager agent has no direct involvement with the environment. The manager agent simply has access to basic GPS coordinates within the environment and has no knowledge of the environment layout, structure and obstacle locations.

The two agent types are arranged in a semi-centralized, peer-to-peer network as seen in Fig. 4. Through the peer-to-peer communication channel, the manager agent distributes target GPS locations to each worker agent currently connected to it. Once given a target destination, worker agents move towards their objective. During their traversal, the worker agents interact with the environment by detecting obstacles with their sensors and controlling their motors to move towards their target.

As seen in Fig. 5, the worker agents have multiple working layers that handle navigational, sensory and computational tasks. There are two main layers: software and hardware. Within the software layer, agent brain handles all of the ongoing communications with other agents and exchanges information with the lower modules: sensor and motor controllers. The sensor controller gathers hardware level information from the distance measuring sensors. It then uses the sensor information to measure distances and direction to obstacles and finds a collision-free path to relay to agent brain for navigation. Once given a collision-free path, the agent brain provides

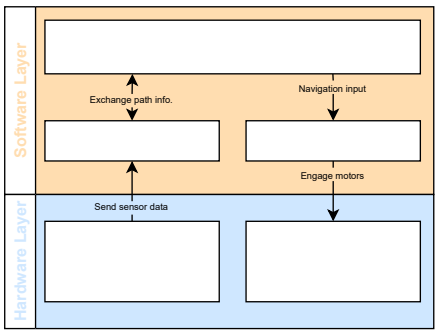

Fig. 5. Worker agent architecture

navigational input (forward, left, right, backward) to the motor controller, which it uses to move the wheels of the vehicle based on the input.

The manager agent periodically asks the worker agents to send it information about themselves and the environment. The worker agents share the following information with the manager agent:

- Current location in GPS coordinates
- Current state (Idle, Navigating, Working)
- Detected obstacle locations in GPS coordinates

Once information on all worker agents have been gathered, the manager agent broadcasts the GPS locations of worker agents back to all of them. The broadcast information can be used by the worker agents when they need to identify each other. They can detect and identify each other by comparing the estimated locations of detected obstacles with the location information of other agents. If the two locations are in close proximity to each other, the detecting worker agent sends a message to the suspected worker agent to acknowledge each other, slow down their speed and make adjustments to their trajectory in order to avoid a possible collision.

Current state information is used by the manager agent to keep track of worker agents' current status. The worker agents who are in an "Idle" state are given a navigation task starting from their current GPS location towards a random GPS location in the environment with no information as to how to get there. The worker agents in the "Navigating" state are given information about other agents' GPS locations periodically to help them avoid colliding with each other. And lastly, the worker agents currently in a "Working" state indicate to the manager agent that the worker agent has reached its target destination and is going to stand there for five seconds before going into the "Idle" state.

Gathered GPS locations of detected obstacles are compiled and visualized by the manager agent to create a virtual map of the working environment. Initially, the manager agent starts with a blank map roughly the size of the working environment (based on edge GPS coordinates of the environment) and fills in the map with the locations of the obstacles given to it by the worker agents.

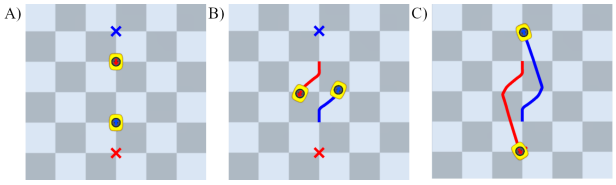

Fig. 6. Case 1, test run from start to finish

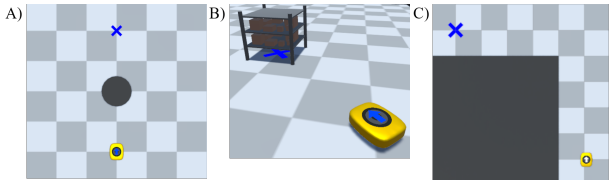

Fig. 7. Starting positions of the agents for Case 2

IV. Experiments & Results

The proposed collision avoidance method has been implemented in a simulation developed using Unity platform [15] in C# programming language [16]. Within the simulation, three test cases have been created. Case 1 tests the agents ability to avoid collisions with other agents. Case 2 tests the agents ability to avoid collisions with various obstacles that cause a local minimum. Case 3 combines the two previous cases in a larger, denser unknown environment where one test run is executed with fuzzy logic and information sharing activated (FLIS), and one test run is executed without them for comparison as a baseline (no FLIS).

1) Case 1 - Agent to agent test: In this test case, the agents' ability to identify and safely avoid collisions with another agent is evaluated.

As seen in Fig. 6 A, two worker agents have been arranged to start from opposite sides in a small test environment where they need to move to the opposite side where the other agent is currently standing in front of. The X markers on the figure denote the target destinations for the two agents. The blue X marker to the north side of the grid is the target destination for the agent to the south side of the grid. The red X marker to the south side of the grid is the target destination of the agent to the north side of the grid. Fig. 6 B shows the agents moving away from each other and Fig. 6 C shows the agents reaching their target destinations. For Case 1, both agents were able to detect, identify and avoid colliding with each other in a head-on collision situation.

2) Case 2 - Various obstacle tests: As part of this test case, three test scenarios have been created to identify the agent's usage of the fuzzy risk estimation to avoid collisions and local minimums. The three scenarios can be seen in Fig. 7. In scenario A, the agent is tasked to move directly behind a large obstacle. In scenario B, the agent is tasked to move underneath a storage shelf from a diagonal side. In scenario C, the agent is tasked to move around a corner.

The results for the three scenarios can be seen in Fig. 8. The agent was able to avoid all three local minimum traps

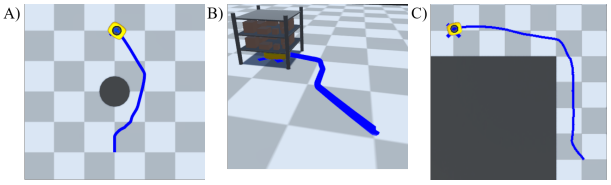

Fig. 8. Results for the three test scenarios of Case 2

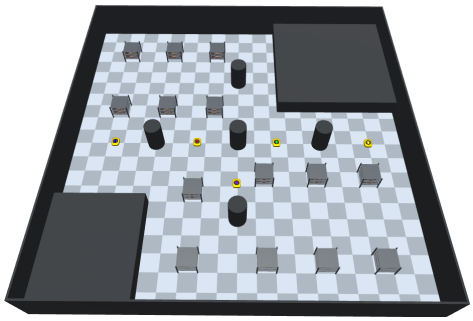

Fig. 9. Simulated test environment for Case 3

and move to its target destinations for all scenarios.

3) Case 3 - Combined test: In the combined case, the previous two test cases have been combined together in a larger and denser environment in which there are five functionally identical worker agents and a single manager agent. The environment houses 14 storage shelves, two corner pieces and five large columns. The simulated environment can be seen in Fig. 9.

As part of the combined test, two separate sessions are held within the environment. One session using the proposed method (FLIS) and another session without using the method (no FLIS). In both sessions, the manager agent distributes random GPS locations as target destinations for the five worker agents repeatedly until the session ends. During the sessions, the number of collisions are recorded. For both sessions, 5, 15 and 30 minute test runs have been conducted where each run was repeated 15 times for accuracy.

The average number of collisions encountered during both test sessions can be seen in Fig. 10. In the figure, "FLIS" represents the test session with fuzzy logic and information

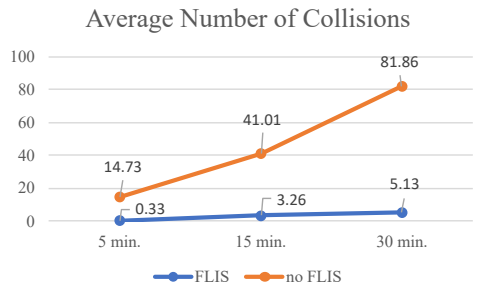

Fig. 10. Average number of collisions for both sessions.

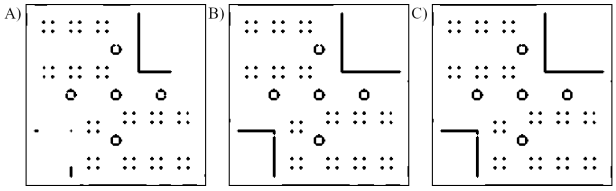

Fig. 11. Virtual map creation progress through 5 minute sessions repeated 15 times

sharing enabled and "no FLIS" represents the session with the method disabled (baseline). With the method enabled, the agents encountered an average of 0.33 collisions for 5 minute runs, average of 3.26 collisions for 15 minute runs and an average of 5.13 collisions for 30 minute runs. With the method disabled, the agents encountered an average of 14.73 collisions for 5 minute runs, average of 41.01 collisions for 15 minute runs and an average of 81.86 collisions for 30 minute runs. Based on the gathered collision information, with the method enabled (FLIS), the agents collided with the shelf legs only. With the method disabled (no FLIS), the agents collided at least once with every type of obstacle in the environment including other agents.

Fig. 11 shows the virtual map creation progress from 5 minute test sessions repeated 15 times. Fig. 11 A shows the snapshot of the virtual map after 5 runs, Fig 11 B shows the virtual map after 10 runs and Fig. 11 C shows the final snapshot of the virtual map. A total number of five agents were able to map the environment in Fig. 9 within 10 test runs which corresponds to 50 minutes in total.

V. Discussion

Based on the results from the initial two test cases (Fig. 6 and Fig. 8), the method allowed AVs to avoid collisions with other agents and obstacles in an unknown environment. The third case results (Fig. 10) show improvement over the baseline (no FLIS). However, the agents, even while using the method, recorded collisions with the shelf legs during the third test case. The collisions with the shelf legs in Case 3 indicates that the results from Case 2 scenario B (Fig. 8 B) do not translate directly over to a multi-agent setting. The inclusion of other AV presence near a challenging local minimum trap may negatively impact the trap escape strategy utilized in the method. Overall, the method was capable of avoiding all collisions with agents and obstacles (other than shelf legs) while also creating a virtual map of the unknown environment.

Compared to the works presented in papers [4]–[6], the method provided collision-free navigation in a multi-agent unknown environment in contrast to a single-agent unknown environment. Compared to papers [7]–[12], the method works in an unknown environment rather than a fully known and observable one. And, similarly to paper [13], the method is capable of creating a virtual map of the unknown work environment using information sharing.

VI. Conclusion and Future Work

In this paper, a collision avoidance method utilizing fuzzy risk estimation and multi-agent information sharing has been presented. The method is designed to work in unknown environments where there is no environmental information available for the AVs to utilize. The fuzzy collision risk estimation is made through mapping sensor and velocity data to a rule base. The AVs use the risk estimation to adjust their speed and current trajectory to avoid collisions as well as local minimum traps. The AVs also exchange information on their current GPS location, state and detected obstacle locations in order to avoid collisions with each other and ultimately create a virtual map of the environment. The method has been implemented and tested within a simulation to evaluate it in agent-to-agent and static obstacle avoidance scenarios which present a local minimum trap. The findings indicate that the method is capable of providing collision avoidance for multiple agents in an unknown environment, while also avoiding the majority of local minimum traps that were tested for.

In the future, the method's local minimum trap avoidance strategy needs improvements to also factor for the presence of other AVs in local minimum proximity. Additionally, the method can be implemented in a real-world setting to evaluate the effectiveness of the method further.

References

[1] H. Ahuett-Garza and T. Kurfess, "A brief discussion on the trends of habilitating technologies for Industry 4.0 and Smart manufacturing," Manufacturing Letters, vol. 15, pp. 60–63, Jan. 2018, doi: 10.1016/j.mfglet.2018.02.011.

[2] L. Schulze and A. Wullner, "The Approach of Automated Guided Vehicle Systems," in 2006 IEEE International Conference on Service Operations and Logistics, and Informatics, Jun. 2006, pp. 522–527. doi: 10.1109/SOLI.2006.328941.

[3] S. A. Deloach, M. F. Wood, and C. H. Sparkman, "Multiagent systems engineering," Int. J. Soft. Eng. Knowl. Eng., vol. 11, no. 03, pp. 231–258, Jun. 2001, doi: 10.1142/S0218194001000542.

[4] E. Marchesini and A. Farinelli, "Discrete Deep Reinforcement Learning for Mapless Navigation," in 2020 IEEE International Conference on Robotics and Automation (ICRA), May 2020, pp. 10688–10694. doi: 10.1109/ICRA40945.2020.9196739.

[5] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sep. 2017, pp. 31–36. doi: 10.1109/IROS.2017.8202134.

[6] H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," IEEE Transactions on Systems, Man, and Cybernetics, vol. 25, no. 3, pp. 464–477, Mar. 1995, doi: 10.1109/21.364859.

[7] G. Kumar T. and V. P. Vijayan, "A Multi-agent Optimal Path Planning Approach to Robotics Environment," in International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), Dec. 2007, pp. 400–404. doi: 10.1109/ICCIMA.2007.228.

[8] S. Bogdan, I. Rigo, and D. Miklic, "Fuzzy logic navigation in multi agent systems," in 2008 IEEE International Symposium on Intelligent Control, Sep. 2008, pp. 1308–1313. doi: 10.1109/ISIC.2008.4635944.

[9] A. Souliman, A. Joukhadar, H. Alturbeh, and J. F. Whidborne, "Real time control of multi-agent mobile robots with intelligent collision avoidance system," in 2013 Science and Information Conference, Oct. 2013, pp. 93–98.

[10] Chia-Han Lin and Ling-Ling Wang, "Intelligent collision avoidance by fuzzy logic control," Robotics and Autonomous Systems, vol. 20, no. 1, pp. 61–83, Apr. 1997, doi: 10.1016/S0921-8890(96)00051-6.

[11] L. Makarem and D. Gillet, "Information sharing among autonomous vehicles crossing an intersection," in 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Oct. 2012, pp. 2563–2567. doi: 10.1109/ICSMC.2012.6378131.

[12] B. Subramanian M, Dr.K.Sudhagar, and G.RajaRajeswari, "A Study on seamless information sharing between robots in identifying the optimal path: An agent based approach," International Conference on Advances in Communication, Network, and Computing (CNC), 2014, pp. 226-235

[13] J. Wang, T. Shan, and B. Englot, "Virtual Maps for Autonomous Exploration with Pose SLAM," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nov. 2019, pp. 4899–4906. doi: 10.1109/IROS40897.2019.8967853.

[14] Y. C. Dundar, "Dynamic path finding method and obstacle avoidance for automated guided vehicle navigation in Industry 4.0," Procedia Computer Science, vol. 192, pp. 3945–3954, Jan. 2021, doi: 10.1016/j.procs.2021.09.169.

[15] "Unity Real-Time Development Platform — 3D, 2D, VR & AR Engine," Unity. https://unity.com (accessed Mar. 29, 2023).

[16] "A tour of C# - Overview," Feb. 13, 2023. https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ (accessed Mar. 29, 2023).