Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 192 (2021) 3945–3954

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Dynamic path finding method and obstacle avoidance for automated guided vehicle navigation in Industry 4.0

Yigit Can Dundar[a,*]

[a] *UiT University of Tromsø, Hansine Hansens veg 18, Tromsø N-9019, Norway*

Abstract

Within the scope of Industry 4.0, Automated Guided Vehicles (AGVs) are used to streamline logistics through the usage of efficient path finding methods. The current path finding methods in the industry rely on excessive usage of guidance in the shape of magnets, tapes or QR codes on the floor that the AGVs follow to reach their destinations. However, the current methods lack operational flexibility and are costly to scale in the cases of job-shop floor expansions. In this paper, a dynamic path finding method with obstacle avoidance is presented which utilizes distance measuring sensors to avoid obstacles and reach the goal destination using the most direct path as possible. Tests for functionality and multi-agent scaling have been conducted to evaluate the performance of the dynamic method in a multi-agent setting. The results show that the dynamic method scales properly and is capable of navigating multiple agents through a simulated warehouse environment autonomously and without relying on external guidance. The dynamic method is able to avoid most collisions using distance measuring sensors and multi-agent negotiation to resolve conflicts among the agents that could have resulted in potential collisions. The proposed dynamic method provides a flexible and scalable path finding method for use in Industry 4.0.

© 2021 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of KES International.

Keywords: Automated Guided Vehicle; Dynamic Path finding; Industry 4.0; Multi-Agent System; Obstacle Avoidance

1. Introduction

Industry 4.0 is an approach to manufacturing that is driven by the need for adapting production to the ever changing needs of the market [10]. A subject of Industry 4.0 that has seen major developments so far is logistics. Current internal and external logistics benefit from automation technologies in the form of Automated Guided Vehicles (AGVs). AGVs require path finding to ensure safe and efficient transportation. AGVs can be managed and organized through the utilization of the Multi-Agent Systems (MAS) approach. With the MAS approach, AGVs can communicate, coordinate and cooperate to handle complex tasks. Techniques like negotiation can also help resolve conflicts regarding

* Corresponding author.
E-mail address: yigit.c.dundar@uit.no

1877-0509 © 2021 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of KES International.
10.1016/j.procs.2021.09.169

resource allocation so that the overall workflow of AGVs are not disrupted or put in a deadlock situation. These types of systems, utilizing AGVs with MAS structures, still require further research and development in order to improve collision avoidance, path finding and conflict resolution.

One of the most labor-intensive and costly activities within warehouses is the order picking process [2]. The current utilization of path finding methods for AGVs depends on mostly static external guidance in the shape of lines, tapes or magnets on the ground which the vehicles scan and follow to reach their destinations and avoid obstacles in an environment. This reliance on external guidance has negative implications when considering scaling costs and reconfigurability [1]. For example, when a layout change is required within the warehouse, the guidance tapes or magnets need additions or adjustments which require resources and additional workforce. In short, guidance reliant approaches lack the flexibility required by Industry 4.0 standards [1].

Contrary to static path finding approaches that rely on external guidance, this paper presents a dynamic path finding method with obstacle avoidance for multi-agent AGV systems that allows the AGVs to work in dynamic environments without relying on excessive external guidance from their environment. The dynamic method is inspired by the Potential Field path finding algorithm's [3] usage of repulsive and attractive forces to find the optimal path. The dynamic method uses distance measuring sensors to allow real-time path finding and obstacle avoidance for AGVs. Allowing the AGVs to move around autonomously imposes another challenge where AGVs can become obstacles for each other. To solve such cases, negotiation and conflict resolution are deployed to ensure that the AGVs can accomplish their tasks without hindering each other. The dynamic path finding method is implemented to run in a virtual simulation that contains 10 AGVs working in a multi-agent environment.

Through this research, the technical details, challenges, strengths and weaknesses of the dynamic path finding method are aimed to be highlighted through testing the dynamic path finding method for functionality and multi-agent scaling. During the tests, the dynamic method's multi-agent scaling and performance metrics are recorded and are then discussed upon. Based on the findings, the dynamic method has the potential to be used in Industry 4.0 with emphasis on manufacturing and logistics related work loads. The proposed method can potentially cut costs related to expansions (warehouse expansions, layout changes etc.) and adapt to ever changing environments autonomously and without requiring external guidance.

2. Background

Industry 4.0 is a strategic initiative to pioneer the ongoing revolution of the manufacturing sector [9]. This revolution was triggered due to developments made in Information and Communication Technologies which allows smart automation of cyber-physical systems with decentralized control alongside advanced connectivity [10]. Through these technological advancements, businesses in Industry 4.0 can adapt to rapid increase and variety in customer demands while also being able to compete with other businesses working in similar areas of production. Resource efficiency and flexibility in production are two important factors that define the need for Industry 4.0 [11]. Within Industry 4.0, advancements in technology like automation, smart factories, self-optimization, self-organization and more open up possibilities for efficient and flexible solutions to be applied. In fact, since the beginning of the Industry 4.0 movement, the logistics sector of production has seen improvements to working efficiency and flexibility through the usage of AGVs [1, 6, 12].

Multi-agent systems approach is suitable for managing and organizing AGVs within the scope of Industry 4.0. Multi-agent systems (MAS) is a field of artificial intelligence where it presents the principles for construction of systems including multiple agents and providing the mechanisms needed for coordination of individual agents [4]. When working with a problem domain that is complex, large or unpredictable, a way to reasonably address the problem is to implement a number of functionally specific and modular components (agents) that each has their own specializations to solve a particular aspect of the problem [13]. These modular components, or agents, work within an environment in which they fulfill their tasks while also communicating and coordinating with each other. The agents can also interact with their environment where they can influence it and can be influenced by it during operation. In multi-agent systems, identifying and resolving different viewpoints and conflicts between agents is a difficult task when agents try to coordinate their actions [13]. In order to resolve conflicts between agents, negotiation techniques can be used where two or more agents assess the situation from their perspective and negotiate a solution [17].

3. Related Work

Path finding is a part of transportation and movement that is concerned with finding the most efficient path from a point in place, to a destination. Path finding, or path planning, is a fundamental requirement for AGVs to work effectively [6]. Hence, path finding research has been conducted for many years and has also been receiving further attention since the beginning of the Industry 4.0 movement. In the past, researchers have looked into the possible path finding methods for AGVs. In one instance, a smart factory prototype has been developed in which the environment is mapped as a 2D grid and the AGVs can utilize the A* path finding algorithm to traverse the environment like it was a graph [5]. The algorithm can generate a path for each AGV on the factory floor with collision and obstacle avoidance. The authors highlight an area for future work where a switch to a dynamic path finding algorithm can improve the multi-agent systems' performance while also being capable of replacing the collision avoidance rules [5]. Compared to the algorithm presented in paper [5], the dynamic method does not require the environment to be mapped and is capable of avoiding moving obstacles as well as static. Dynamic path finding research also corresponds to the future work that the authors of paper [5] have identified.

In another paper, an altered version of the Dijkstra path finding algorithm, that takes traversal time into account, has been used for AGV path finding to traverse a 2D graph of a simulated environment [6]. The work provided by the authors of paper [6] requires a 2D graph-like representation of an environment. Although sufficient for static environments, their algorithm may not be suitable for use in dynamic environments. Also, the authors made no assumptions regarding the presence of obstacles and how to avoid them in their paper, which the dynamic method that is presented in this paper takes into consideration when planning a path to a destination for AGVs.

Another paper [8] presents a multi-agent path finding solution that utilizes priority based optimization where the agents need to pay attention to their battery levels as well as avoiding collisions with obstacles and with other agents in the environment. The method presented in the paper relies on the pre-processing of the environment in order to provide the agents with the optimal path based on resource allocation priorities. The authors of paper [8] plan to extend their studies with experimental evaluations to understand the scalability of their method. Compared to paper [8], the dynamic path finding method, that this research paper presents, does not rely on pre-processing the multi-agent environment but instead uses real-time collision avoidance and negotiation to provide the optimal path for the agents.

Another research presents a different approach to path finding by utilizing the Potential Field path finding algorithm to generate artificial potential function of the environment using distance transforms [7]. In the Potential Field algorithm, a global representation of an environment can be obtained in order to plan a path at the global level [3]. The algorithm can generate a representation of an environment as a field of vectors flowing from a starting position, to a goal destination. The goal destination acts as a sink and generates an attractive force that directs vectors towards it, and the obstacles generate repulsive forces where they direct the vectors away from them. The authors of paper [7] have implemented the algorithm using an AGV on a small scale to work in a custom environment with obstacles and destinations. The findings of paper [7] indicate that the potential field method for path planning can save pre-processing time compared to other methods. Similar to the approach presented in paper [7], the dynamic path finding method also relies on the usage of attractive and repulsive force vectors to avoid collisions and provide path finding for AGVs. Unlike the approach in paper [7], the dynamic method handles the calculations of these force vectors during run-time instead of pre-processing them.

4. Dynamic Path Finding

The dynamic path finding method, presented in this paper, is inspired by the usage of repulsive and attractive forces found in the Potential Field path finding algorithm. The obstacles in the environment repel the agent from their direction and the goal destination pulls the agent towards it. The Potential Field path finding algorithm was not originally designed to work in real-time, multi-agent and dynamic environments. It works in static environments to find an optimal path, before run-time, using pre-processing to calculate the attractive and repulsive vector fields. However, with some modifications to the way the Potential Field algorithm works, the usage of repulsive and attractive vector forces can be calculated for multi-agent and dynamic environments in real-time. The dynamic path finding method uses proximity sensors that can measure distances to objects to calculate a part of the path instead of calculating it all

at once. This allows the agents to adapt to rapid changes in the environment especially when the environment to be traversed is not entirely known to the agent.

The dynamic method uses both the direction towards the goal destination and the summation of the directions away from obstacles based on the agents current position in the environment to find the optimal path. The direction towards the goal destination, the attractive force vector, always influences the path of the agent whereas obstacles only affect it when they are detected by the agent's proximity sensors and are in close range.

$$Dir_{att} = \hat{V_{target}} - \hat{V_{start}} \tag{1}$$

To calculate the attractive force (Dir_{att}), the normalized starting vector for the vehicle ($\hat{V_{start}}$) is subtracted from the normalized target vector towards the goal destination ($\hat{V_{target}}$) as seen in Equation 1. When obstacles are detected by the sensors, first, the agent has to get the distance measurements to the obstacles from the agent's current position.

$$Dist(\vec{O}, \vec{V}) = \sqrt{(\vec{O_x} - \vec{V_x})^2 + (\vec{O_y} - \vec{V_y})^2 + (\vec{O_z} - \vec{V_z})^2} \tag{2}$$

The distance between the obstacle and the agent vehicle is calculated as presented in Equation 2. In the equation, vector $\vec{O}$ refers to the obstacle vector and vector $\vec{V}$ refers to the vehicle vector. If the distance ($Dist(\vec{O}, \vec{V})$) is in critical range, the repulsive force needs to be calculated in order to avoid a potential collision. The critical distance for collisions is set to be roughly half the maximum distance the sensors can detect objects. The exact value for the critical distance is 4 units (points in coordinates) where the maximum range of the sensor is 7 units. These values are assigned based on the vehicle length which is 4 units.

$$Dir_{safe} = \hat{V} - \hat{O} \tag{3}$$

To calculate the repulsive force, the method finds a safe direction (Dir_{safe}) away from the obstacle by subtracting the normalized obstacle vector ($\hat{O}$) from the normalized vehicle vector ($\hat{V}$) as seen in Equation 3.

$$\begin{aligned} &M = Dist_{crit} - Dist(\vec{O}, \vec{V}) \\ &0.5 < M < Dist_{crit} \end{aligned} \tag{4}$$

The safe direction vector then needs to be multiplied by a value (M) which is acquired by subtracting the distance to the obstacle ($Dist(\vec{O}, \vec{V})$ from the determined critical distance ($Dist_{crit}$) for avoiding collisions (Equation 4). The distance value $Dist_{crit}$ is used to determine whether an obstacle is in collision range or not (closer than 4 units). The multiplier M is clamped between the value 0.5 and $Dist_{crit}$ (Equation 4). Therefore, the closer the obstacle is to the vehicle, the greater the multiplier. After the multiplier value is calculated, the safe direction (Dir_{safe}) calculated earlier is multiplied by the calculated multiplier value (M) to generate the final repulsive force vector away from the obstacle (Equation 5).

$$Dir_{rep} = M.Dir_{safe} \tag{5}$$

The agent is responsible for calculating the repulsive force vectors for each obstacle currently in proximity. For every obstacle in critical range, the dynamic method sums up all of the calculated safe direction vectors away from obstacles (Equation 6).

$$\sum_{i=1}^{n} Dir_{safei} = Dir_{safe1} + Dir_{safe2} + ... + Dir_{safen} \tag{6}$$

In Equation 6, the variable n represents the total number of obstacles currently in critical range. After the safe directions are summed up, the average distance (Avg_{dist}) between all obstacles in critical range needs to be calculated (Equation 7).

$$\begin{aligned} &Avg_{dist} = ((\sum_{j=1}^{n} Dist(\vec{O}, \vec{V})_j)/n)/Dist_{crit} \\ &1 < Avg_{dist} < Dist_{crit} \end{aligned} \tag{7}$$

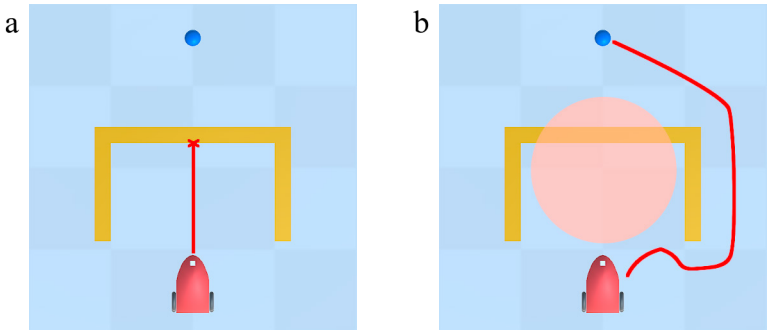

Fig. 1. (a) Local minimum trap example; (b) Resolving local minimum with a soft obstacle

The average distance value is then used to multiply the summation of safe direction vectors to create a final collective direction ($Dir_{repfinal}$) away from multiple obstacles in critical range (Equation 8).

$$Dir_{repfinal} = Avg_{dist} \cdot \sum_{i=1}^{n} Dir_{safei} \tag{8}$$

Finally, as seen in Equation 9, the method sums up the direction vectors (towards the goal and away from obstacles) and adds them to the current position vector of the agent to provide it an arbitrary path and direction that it needs to follow to reach its goal while avoiding obstacles in its path.

$$TargetPath = V_{start} + (C.Dir_{att}) + Dir_{repfinal} \tag{9}$$

In Equation 9, the reason why Dir_{att} is multiplied by a coefficient C is to push more incentive towards going to the goal destination at all times. The coefficient can be increased or reduced to alter the attractive force being applied to the agent. It is ideal to keep it above 1.0 to make sure that the path finding method prioritizes going towards the goal at all times. In the simulation, the coefficient C is assigned 2.0 as its value. All of the above calculations are executed every 0.2 seconds within the simulation until the agent reaches its target destination.

A common challenge encountered when using the potential field path finding algorithm is the local minimum issue [3, 14]. The local minimum issue in potential field path finding algorithm is often observed when an agent needs to go to a destination directly behind a large object. In such situations, both the attractive and repulsive vector fields that affect the agent overlap with each other and the calculated vector value pushes the agent to collide directly with the obstacle. Since the dynamic method utilizes attractive and repulsive vectors, the local minimum issue is present for it as well. This issue requires a solution, otherwise, collision with obstacles, in local minimum situations, becomes highly likely. To get around this issue, the dynamic path finding method records the coordinates for the local minimum area and marks it as a soft (virtual) obstacle in memory when an agent detects such an area in the environment. An agent detects the local minimum if the arbitrary path that it calculated points directly towards the goal and there is an obstacle right in front of it. When the agent encounters such a path, it will try to move away for 5 seconds and if the path towards collision still persists, the area is marked in memory as a local minimum. The marked soft obstacle represents a small area that the agent needs to calculate repulsive forces from and avoid in the future. The local minimum location information is removed from memory once the agent is able to move away from the marked area.

In Fig. 1 (a), the path for the agent leads it on a direct collision course with the obstacle in front of its goal destination (denoted as the blue dot). The reason why this happens is that the attractive force cancels the repulsive force due to both of them overlapping, and since the attractive force is always higher than any repulsive force, the agent is forced to move directly towards the goal destination which will inevitably result in a collision.

In Fig. 1 (b), the agent marks the exact position it recognized as a local minimum and creates a virtual soft obstacle (orange circle) representation of the area in memory. The agent then adjusts its attractive and repulsive forces to escape the local minimum to reach its goal destination. Stochastic elements in real-life environments can cause local minimum traps to occur dynamically. For example, if an item falls off of a storage shelf down on to the floor, the

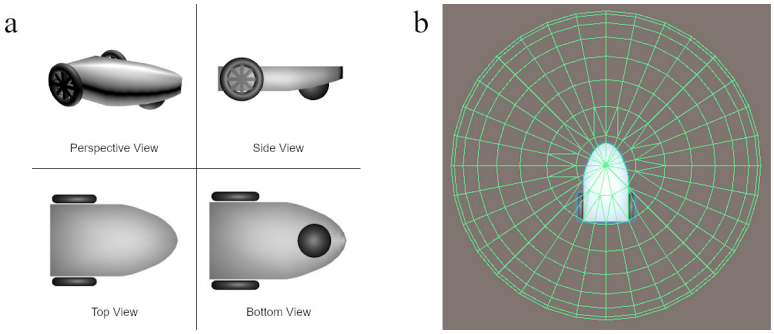

Fig. 2. (a) Agent vehicle from different viewing angles; (b) Proximity sensor radius of the vehicle for the simulation

AGVs may be blocked off from the path that they were set on. In such cases, local minimum escape solution provided by the dynamic path finding method can help the AGV find a way around the obstacle to continue its traversal.

The dynamic path finding method gives independent responsibility to the agent that is traversing the environment to reach its destination without colliding with obstacles both static and dynamic. The agent detects obstacles using its proximity sensors in a 360 degree radius with distance information. Most obstacles, that affect the path of the agent, are mostly detected towards the front of the vehicle so, the 360 degree radius for the sensors is not entirely necessary for proper functionality. The agent uses the distance information to the obstacle to adjust its rotation to an arbitrary path which it follows to reach its final destination. This arbitrary path is the summation of the repulsive and attractive forces the agent is currently under the influence of and the path is represented by a short distance away from the agent towards the calculated, collision free, direction. During traversal, the agent can encounter local minimum traps in which it is forced to collide head-on with an obstacle standing directly in front of its goal destination. To resolve this issue the agent records the location of the local minimum and creates a virtual, soft obstacle in memory from which it starts calculating repulsive forces. This solution can help the agent escape common local minimum trap situations and allow it to complete its traversal of the environment without colliding with obstacles in the environment.

5. Multi-Agent Environment

As part of the dynamic path finding method implementation, a multi-agent environment was created to simulate a storage warehouse environment. Inside this environment, there are storage shelves with work areas spread around their sides where two distinct agent types, worker agents and the manager agent, communicate and cooperate with each other to fulfil work orders at those work areas. The worker agents utilize the dynamic path finding method to move from work area to work area while the manager agent gives the worker agents work orders that they need to fulfil. However, work order conflicts can occur between the worker agents, where they may have work orders that take place in the same work areas. These conflicts need to be resolved in order to avoid collisions and potential deadlocks. The following sections provide further detail into the two agent types and how such conflicts are resolved in the simulated multi-agent environment.

5.1. Worker Agents

The worker agents are the main path finding agents in the simulated environment. These agents have continuous work tasks which they need to move towards while avoiding static and dynamic obstacles (other agents). The work tasks are simply to go over to a specific spot around a shelf, wait there for a specific amount of time and move to a different work area afterwards.

The shape of the vehicle, that the agents drive, can be seen in Fig. 2 (a). This shape allows the agent to recover easily from frontal collisions, in case they occur, thus allowing the agent to continue working after a frontal collision. The two-wheel setup is also simple to control by the agent and allows it to move and turn to any direction in the horizontal axis (relative to the agent). The wheels also allow independent control which is used when turning the vehicle left or right. The spherical object attached to the frontal side of the vehicle's bottom is a low friction static

object that keeps the front of the vehicle up in order to balance the natural stance of the vehicle. The vehicle is also accompanied by sensors around its chassis to cover a 360 degree area in the horizontal axis with a radius size of 7 units around a 4 unit long vehicle.

In Fig. 2 (b), the green circle with the origin point located around the front of the vehicle represents the general area of the sensor through which obstacles can be detected. The sensor is capable of detecting both moving and stationary objects within its range and also provides distance information to individual, in-range, obstacles. Thus, using the distance information, the agent can then calculate repulsive and attractive forces which the dynamic path finding method requires.

The worker agents can switch between a total of four different states during run-time. These states allow the agent to organize its behavior both towards achieving its goals and towards negotiating with other agents to resolve work order conflicts. The four states of the worker agents are:

- Idle state: In this state, the agent is waiting for a task to be assigned to it by the manager agent.
- Goal state: In this state, the agent has a goal destination determined to it accompanied by a work order that it needs to handle at that location. The agent stays in this state until it reaches its goal destination.
- Work state: In this state, the agent has reached its goal destination and is now carrying out the work order that it was given.
- Negotiation state: This special state is entered when work order conflicts arise between the agents. While in this state, the worker agents need to wait for the manager agent to give them further instructions as to how to resolve the conflict.

The worker agents switch between these four states during run-time, continuously in order to stay productive. There is no exit state, the agents stop working when the simulation is terminated.

5.2. Manager Agent

The manager agent is a singular, meta-agent that is responsible for assigning work orders to worker agents and keeping track of their progress. This agent does not have a physical presence in the environment and is reliant on back-and-forth communication between the worker agents and itself. The manager agent assigns random work orders, with varying priorities, to each and every idle worker agent. Due to the nature of random assignments, two, or more, worker agents can be responsible for the same work order. The manager agent is responsible for resolving such conflicts through negotiations held between itself and the worker agents that are in a conflict situation.

Before negotiations can take place, the conflict needs to be identified. The worker agents are not aware of other worker agents' whereabouts or their work orders. So, the worker agents require assistance from the manager agent to detect if other agents require access to the same work area around the same time. A conflict is identified when the manager agent detects that two or more worker agents are all in close proximity of a work area that they all require access to at the same time. Once identified, the conflict needs to be resolved through negotiation between the manager agent and the worker agents. First, the manager agent commands all worker agents, involved in the conflict, to go into their negotiation state where they wait for further instructions from the manager agent. Second, the manager agent requires the worker agents to present their work order priorities. Then, a queue is created where the agent with the highest priority stands at the head of the queue and the rest of the agents are ordered in descending order based on their priorities. After the queue has been created and ordered based on priority, the manager agent prompts the agent at the head of the queue with a message saying that it can proceed to the work area. And, once that agent finishes its work order inside that work area, the manager agent removes that agent from the work queue and prompts the next head of the queue to proceed until the queue is empty. The negotiation concludes when there are no agents left in the queue. There is another special case of conflict that can arise after a negotiation round had already started. In this special case, a late agent approaches a work area that already has a queue designated for it in order to resolve the conflict. To resolve the special conflict, the late agent is put to the end of the queue regardless of its priority. The rest of the process is the same afterwards. Through this negotiation, work order conflicts among worker agents are resolved by the manager agent which helps avoid potential deadlocks and collisions.

Manager agent acts as a centralized command center during negotiations which can cause potential bottlenecks and single point of failure situations. To overcome such issues, a backup manager agent can be put in place in the cases

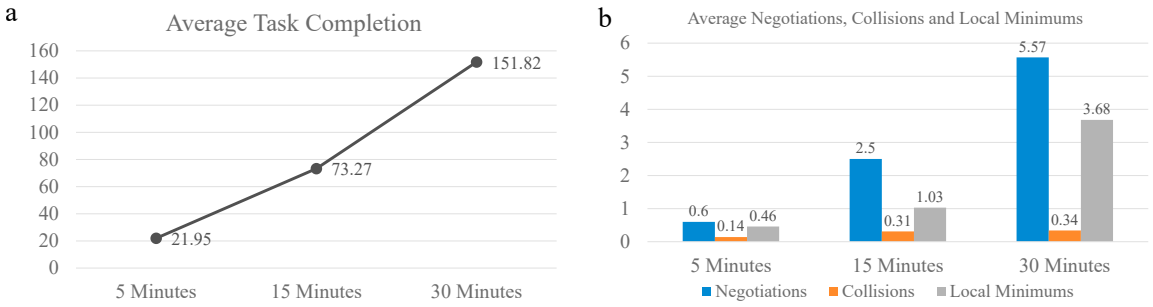

Fig. 3. (a) Average task completion rate; (b) Average number of negotiations, collisions and local minimums

where the original manager agent failed. A move towards a decentralized negotiation structure where worker agents negotiate entirely on their own can also be a potential solution to issues that may arise from the centralized approach to negotiation.

6. System Performance and Discussion

The dynamic path finding method and the multi-agent environment were virtually simulated using the Unity Platform [15] where the code was written in C# programming language [16]. Inside the simulated environment, there are 20 storage shelves (with four work areas for each), 10 worker agents and one manager agent.

To test the system for functionality, three test runs were conducted all lasting different time intervals: 5 minutes, 15 minutes and 30 minutes. These tests were repeated 50 times to gather average values for task completions, negotiations, collisions and local minimums. The reason why the tests are repeated is due to the randomized nature of work order distributions among worker agents. Since each test run would be different from the ones before or after, repetition helps cover most of the possible random combinations which represents the system performance more accurately. The average task completion rates for the three test cases can be seen in Fig. 3 (a).

In Fig. 3 (a), the Y-axis represents the average number of tasks completed and the X-axis represents the three test cases: 5, 15 and 30 minute runs. The worker agents were able to complete 21.95 work orders on average in 5 minutes, 73.27 work orders on average in 15 minutes and 151.82 average work orders in 30 minutes. The average number of negotiations, collisions and local minimums can be seen in Fig. 3 (b). In Fig. 3 (b), the Y-axis represents the average numbers for negotiations, collisions and local minimums whereas the X-axis represents the three test cases for 5, 15 and 30 minutes respectively. The agents negotiated an average of 0.6 times for 5 minute runs, average of 2.5 times for 15 minute runs and an average of 5.57 times for 30 minute runs. Some collisions were also recorded by the worker agents. An average of 0.14 collisions were recorded in 5 minute runs, an average of 0.31 collisions were recorded in 15 minute runs and an average of 0.34 collisions were recorded in 30 minute runs. Average recorded local minimums are as follows: 0.46 local minimums in 5 minutes, 1.03 in 15 minutes and 3.68 in 30 minutes.

The dynamic path finding method was also tested to examine its multi-agent scaling performance. The method has been tested using three separate agent configurations: single agent, five agents and 10 agents. During the tests for the three agent configurations, the total task completion rate and the average task completion rate per agent were recorded in 15 minute long tests which were repeated 50 times for recording accurate results. The results for the multi-agent scaling tests can be seen in Fig. 4.

In Fig. 4, the Y-axis represents the number of work orders completed and the X-axis represents the agent configurations: single agent, five agents and 10 agents. A single agent was able to complete a total of 8.3 work orders on average. Five agents were able to complete 39.2 work orders in total and each agent, individually, completed 7.84 work orders on average. Finally, 10 agents completed 73.27 work orders in total and individually, each agent within the 10 agent configuration completed 7.32 work orders on average.

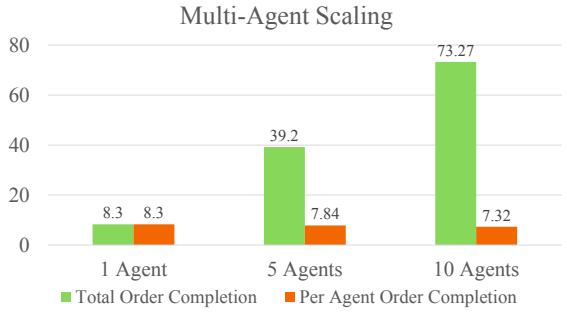

Fig. 4. Multi-agent scaling based on total and per agent work order completion rates

6.1. Discussion

Based on the results, the dynamic path finding method, based on the Potential Field algorithm, is able to navigate multiple agents through a dynamic environment in real-time using sensors and negotiation. Unlike the original Potential Field algorithm which was meant to be used in static, single-agent environments. The average task completion rates for the three time intervals show proper scaling, as seen in Fig. 3 (a), in an environment prone to work order conflicts and possible deadlocks. These results highlight the importance of negotiation in such settings based on the findings represented in Fig. 3 (b) where the number of negotiations scale similarly to the tasks completed. This indicates that, without negotiation, the task completion rates might not have scaled properly. In terms of collision avoidance, the dynamic path finding method was not able to avoid some collisions as seen in Fig. 3 (b). This outcome could be due to the the dynamic method not applying enough or appropriate repulsive forces in high-traffic scenarios where multiple agents pass through the same area at the same time. The amount of local minimums scale similarly to the amount of tasks completed per test case. This can also factor in as to how the task completion rate scales properly. Since, escaping local minimum traps allowed the agents to continue moving towards their destinations unhindered. Compared to related path finding methods, the dynamic method is able to provide AGVs with path finding and obstacle avoidance capabilities without pre-processing the environment and without using external guidance. The dynamic method also shows proper multi-agent scaling as seen in Fig. 4. As the number of agents increases, the average work order completion rate per agent drops slightly while the total work order completion rate increases as expected.

7. Conclusion and Future Work

Through this research, a dynamic path finding method with obstacle avoidance has been presented for AGVs utilized in Industry 4.0. Unlike some of the solutions deployed in the industry, this method does not rely on excessive external guidance (magnets, tapes, QR codes on the floor etc.) to provide path finding capabilities for AGVs. Instead, through the usage of proximity sensors, the dynamic method can detect and avoid obstacles by calculating the attractive and repulsive forces, found in the Potential Field path finding algorithm, in real-time. The method as well as the multi-agent environment have been implemented to run in a virtual simulation to test for functionality and multi-agent scaling. The results indicate that the overall system works properly with the help of multi-agent negotiation to resolve agent conflicts and scales properly as the number of agents increase. Negotiation was proven to be an important factor in ensuring that the agents work in an orderly fashion in the cases of conflicts like needing access to a work area at the same time. The usage of a manager agent was also beneficial in organizing worker agents and conducting the negotiation rounds. The manager agent does impose a centralized structure, which may require work around solutions to avoid possible bottlenecks and single point of failure situations. The dynamic method is capable of avoiding most collisions, however, high traffic scenarios caused the collision avoidance to not work properly at times which may require smarter and more aggressive usage of repulsive forces to avoid obstacles. Local minimum escape implementation also shows promise where it helped agents escape trap situations and continue working without getting stuck in places.

The dynamic path finding method has been proven to function and scale properly, and it can potentially benefit future implementations of AGV oriented systems to be used in Industry 4.0. The dynamic method also acts as a foundation for future work in which further research can be conducted. Obstacle avoidance of the dynamic path finding method will be worked on in order to improve robustness in high-traffic scenarios against collisions. There are plans to implement the dynamic path finding method in a real-life environment using a decentralized and distributed architecture to evaluate its feasibility and efficiency with regard to Industry 4.0 standards. Additionally, the dynamic method requires additional testing using complex tasks for the agents which require extensive collaboration and communication between the agents.

References

[1] Theunissen, J., Xu, H., Zhong, R.Y., Xu, X., 2018. Smart AGV System for Manufacturing Shopfloor in the Context of Industry 4.0, in: 2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP). Presented at the 2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), pp. 1–6. https://doi.org/10.1109/M2VIP.2018.8600887

[2] de Koster, R., Le-Duc, T., Roodbergen, K.J., 2007. Design and Control of Warehouse Order Picking: A Literature Review. European Journal of Operational Research 182, 481–501. https://doi.org/10.1016/j.ejor.2006.07.009

[3] Hwang, Y.K., Ahuja, N., 1992. A Potential Field Approach to Path Planning. T-Ra.

[4] Stone, P., Veloso, M., 2000. Multiagent Systems: A Survey from a Machine Learning Perspective. Autonomous Robots 8, 345–383. https://doi.org/10.1023/A:1008942012299

[5] Gochev, I., Nadzinski, G., Stankovski, M., 2017. Path Planning and Collision Avoidance Regime for a Multi-Agent System in Industrial Robotics. Machines. Technologies. Materials. 11, 519–522.

[6] Qing, G., Zheng, Z., Yue, X., 2017. Path-Planning of Automated Guided Vehicle Based on Improved Dijkstra Algorithm, in: 2017 29th Chinese Control And Decision Conference (CCDC). Presented at the 2017 29th Chinese Control And Decision Conference (CCDC), pp. 7138–7143. https://doi.org/10.1109/CCDC.2017.7978471

[7] Wu, K.-H., Chen, C.-H., Ko, J.-M., Lee, J.-D., 1999. Path Planning and Prototype Design of an AGV. Mathematical and Computer Modelling 30, 147–167. https://doi.org/10.1016/S0895-7177(99)00171-5

[8] Bogatarkan, A., Erdem, E., Kleiner, A., Patoglu, V., 2020. Multi-modal Multi-agent Path Finding with Optimal Resource Utilization, in: Wang, L., Majstorovic, V.D., Mourtzis, D., Carpanzano, E., Moroni, G., Galantucci, L.M. (Eds.), Proceedings of 5th International Conference on the Industry 4.0 Model for Advanced Manufacturing, Lecture Notes in Mechanical Engineering. Springer International Publishing, Cham, pp. 313–324. https://doi.org/10.1007/978-3-030-46212-3_24

[9] Xu, L.D., Xu, E.L., Li, L., 2018. Industry 4.0: State of the Art and Future Trends. International Journal of Production Research 56, 2941–2962. https://doi.org/10.1080/00207543.2018.1444806

[10] Rojko, A., 2017. Industry 4.0 Concept: Background and Overview. International Journal of Interactive Mobile Technologies (iJIM) 11, 77–90.

[11] Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., Hoffmann, M., 2014. Industry 4.0. Bus Inf Syst Eng 6, 239–242. https://doi.org/10.1007/s12599-014-0334-4

[12] Mehami, J., Nawi, M., Zhong, R.Y., 2018. Smart Automated Guided Vehicles for Manufacturing in the Context of Industry 4.0. Procedia Manufacturing, 46th SME North American Manufacturing Research Conference, NAMRC 46, Texas, USA 26, 1077–1086. https://doi.org/10.1016/j.promfg.2018.07.144

[13] Sycara, K.P., 1998. Multiagent Systems. AI Magazine 19, 79–79. https://doi.org/10.1609/aimag.v19i2.1370

[14] Kovács, B., Szayer, G., Tajti, F., Burdelis, M., Korondi, P., 2016. A Novel Potential Field Method for Path Planning of Mobile Robots by Adapting Animal Motion Attributes. Robotics and Autonomous Systems 82, 24–34. https://doi.org/10.1016/j.robot.2016.04.007

[15] Unity Technologies, Unity Real-Time Development Platform — 3D, 2D VR & AR Engine [WWW Document]. URL https://unity.com/ (accessed 4.5.21).

[16] Microsoft Corporation, A Tour of C# - C# Guide [WWW Document]. URL https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ (accessed 10.21.20).

[17] Håkansson, A., Hartung, R., 2014. An Infrastructure for Individualised and Intelligent Decision-making and Negotiation in Cyber-physical Systems. Procedia Computer Science, Knowledge-Based and Intelligent Information & Engineering Systems 18th Annual Conference, KES-2014 Gdynia, Poland, September 2014 Proceedings 35, 822–831. https://doi.org/10.1016/j.procs.2014.08.248