Journal of Robotics Research (JRR)
Volume 1, Issue 1, Date: 6 - September - 2024

Advanced Path Tracking and Traffic Management Using IR Sensors and Timed Automata

Atanu Shuvam Roy [1*], Arya Das[2]
[1] Department of Computer Science & Engineering, Indian Institute of Technology, Kanpur U.P. 208016, India.
[2] Department of Aerospace Engineering, Indian Institute of Technology, Kanpur U.P. 208016, India.
Email: [1*] atanusroy22@iitk.ac.in, [2] aryad22@iitk.ac.in
*Corresponding Author

***Abstract*— Path tracking for Automated Guided Vehicles (AGVs) is a critical challenge, particularly in environments with multiple AGVs sharing the same pathways. This challenge becomes increasingly significant in traffic management systems, where efficient coordination and movement are essential to prevent congestion and ensure safety. As the world progresses towards various levels of automation, exemplified by automated delivery robots, the importance of robust AGV path-tracking solutions has escalated. This paper explores existing innovative strategies through review and then attempts to simulate how to mitigate the problem by integrating timed automata and sensors to minimize waiting times, reduce congestion, and im- prove urban traffic system efficiency. Simulation results in Coppeliasim VREP demonstrate that AGVs maintained normal to moderate speeds (5 to 7 units) in high-congestion scenarios, reduces maximum congestion 20% ensuring continuous flow and preventing total blockage.**

Keywords*— *Automated Guided Vehicles (AGVs), path tracking, traffic management, automated intersections, urban traffic control

I. INTRODUCTION

In today's industrial and logistics sectors, the integration of automated guided vehicles (AGVs) has revolutionized operations by automating tasks, enhancing efficiency, and providing the flexibility needed to adapt to various demands. These vehicles have become essential tools for modernizing and streamlining workflows.[1] These autonomous systems are essential in warehouse automation, material handling, and manufacturing processes, significantly boosting productivity while reducing labor costs. AGVs are adept at performing various tasks, from transporting goods and raw materials to managing intricate workflows in dynamic environments. [2] Their deployment has been shown to streamline operations, mitigate human error, and enhance safety in industrial settings. Despite these advantages, path tracking remains a critical challenge, especially in environments with multiple AGVs operating concurrently [3]. Coordinating numerous AGVs within a shared space requires sophisticated path planning and traffic management strategies to ensure seamless and efficient operations. Without effective planning, issues such as path conflicts, deadlocks, and inefficient waiting times can arise, leading to bottlenecks and diminished throughput. These challenges can undermine the benefits of AGV systems. Navigating AGVs through dynamic and complex environments requires not just efficient routing but also advanced coordination strategies to ensure smooth and safe operations. As AGVs become more prevalent in various industries, the need for robust methods to optimize their movement grows. Traffic management plays a key role in this context, focusing on the synchronization of multiple AGVs to avoid conflicts and maintain uninterrupted workflow. Techniques such as scheduling algorithms, priority rules [4], and conflict resolution protocols [5] are critical in managing traffic flow and enhancing system efficiency. Effective traffic management reduces the risk of deadlocks—situations where AGVs obstruct each other's paths—and minimizes waiting times, thereby improving the overall throughput of the system. This paper presents a novel approach to managing congestion at traffic intersections using timed automata, providing an innovative solution for controlling AGV traffic in complex settings.

The subsequent sections will describe the literature review in section 2 and methodology in section 3 followed by results and discussion in section 4 including limitations. Finally, the paper concludes with section 5.

II. LITERATURE REVIEW

In the field of AGVs, significant advancements have been made in enhancing their efficiency and performance. [6] Many studies have explored the PID tuning for AGVs, providing insights into the simulation methods used to enhance their control systems. [7]. And applications which can be deployed in AGV are also on the rise e.g. using TurtleBot2i and RAZBOT AGV platforms into a 3D Unity environment, controlled via ROS, demonstrating accurate simulation and control in both environments [8], analysis of ammonia detection methods in agriculture [9], food-delivery robots [10], salient object detection and 3D object detection in robots [11], farming practices [12] etc.

Moshayedi et al. [13] in their paper evaluate AGV per- formance by simulating and comparing four PID controller tuning methods (Ziegler Nichols, empirical, PSO, BAS) on various paths in MATLAB and CoppeliaSim, finding PSO performs best overall, and BAS is the fastest. In many papers, CoppeliaSim has been used to simulate propositions, for

Received: 8-8-2024
Revised: 3-9-2024
Published: 6-9-2024

example, to enhance Omni robot navigation accuracy and efficiency by analyzing performance metrics over various paths with SLAM.

CoppeliaSim has been used, highlighting the need for optimized speed management to reduce errors and improve safety in diverse applications [14], [9]. Research has been done on other PID tuning methods as well [15].

In modern factories, AGV robots rely on precise navigation for safety, energy management, and adherence to predetermined paths. An effective fusion method combining vision (camera) and infrared (IR) sensors with minimal sensor usage can be employed in this scenario. Researchers [16] implemented this method, which was then simulated and evaluated using the VREP simulator and the Python API. The process was tested on five complex paths, demonstrating superior path tracking at maximum speeds compared to traditional vision-based methods.

In 2021, a time automata-based reliability detection method was proposed to ensure the reliable operation of AGVs.[17] This method involves building a model during the design stage and using timed automata to qualitatively and quantitatively test the model's reliability through iterative simulations. The results demonstrate the method's effectiveness in calculating reliability. Yue et al. [18] employed a model for an improved rule-based heuristic algorithm, integrating Dijkstra and Q-Learning algorithms for optimal scheduling and path planning. Additionally, a new conflict avoidance strategy based on graph theory was introduced to reduce AGV path conflicts. Numerical experiments demonstrated the model and algorithm's effectiveness compared to existing methods.

In 2023, Feng et al. [19] proposed an innovative path planning and trajectory tracking scheme, incorporating static and dynamic obstacle considerations through three potential fields and an optimized model predictive control (MPC) cost function. Enhanced by a fuzzy logic system for dynamic weight adjustments, the scheme includes a fuzzy linear quadratic regulator for lateral control and a PI controller for longitudinal tracking, demonstrating effectiveness in multi-scenario simulations.

The development of intelligent logistics and multi-AGV (Automatic Guided Vehicle) systems has expanded the use of automated warehouses, yet efficient path planning remains a significant challenge. Researchers present an improved A-star algorithm that incorporates current and future congestion costs to predict and avoid congestion in narrow lanes [20]. Simulation comparisons demonstrate the algorithm's potential in reducing runtime conflicts and alleviating traffic congestion. Simge et al [21] in their paper discusses effectiveness of multiple AGVs picking up and delivering orders. It explores warehouse order picking, balancing travel distance, energy consumption, and investment costs. A new algorithm and mathematical model proposed in the paper efficiently reduces travel distances and improves operational efficiency.

Lee et al [22] introduces a novel zone-control algorithm that partitions AGV guide paths into zones based on AGV geometry and guide path topology, effectively preventing collisions and deadlocks, and demonstrating 58-85% performance improvements in handling delivery tasks over state-of-the-art methods in irregular layouts.

Wang et al. in their study enhances the BP (Back Propagation) network algorithm for intelligent AGVs in ports, developing an AGV road sign recognition and visual system, and demonstrates through experiments that the improved algorithm achieves high recognition accuracy and response speed, confirming its practical application value in complex port conditions [23]. Verma et al. [24] introduces an Improved Dynamic Resource Reservation (IDRR) method for AGV systems, which enhances time efficiency and ensures deadlock-free operations by utilizing dynamic multiple reservations of shared resource points, combined with conflict detection and resolution, and demonstrates its effectiveness in productivity, travel distance, and task completion time through extensive simulations. Many works propose using sensors to conduct path tracking and following.

Yang et al. [25] proposes a multi-AGV tracking system integrating a multi-AGV scheduling system, AprilTag, improved YOLOv5 with oriented bounding box (OBB), extended Kalman filtering (EKF), and global vision to efficiently calculate coordinates and heading angles of AGVs with higher positioning accuracy and less time complexity than traditional methods. Sensor fusion-based works have also proved beneficial [26]. Researchers [27] present an enhanced genetic algorithm for multi-AGV path planning, introducing three- exchange crossover operators for better offspring generation and double-path constraints to minimize both total and individual AGV path distances. Simulation results confirm the algorithm's effectiveness in reducing overall and longest path distances.

Liu et al. [28] addresses the challenges of resource allocation, conflict, and deadlock in multi-AGV systems by establishing a scheduling system using a unidirectional directed graph and the A* algorithm for path planning. The system is implemented through programming and validated with a 20-AGV simulation, demonstrating effective conflict resolution, stability, and real-time performance. It offers significant potential for ap- plication in similar multi-AGV scheduling systems. Hassan et al. [29] presents a global off-line path planning approach for Multi-Robot Systems (MRSs) using an energy-based Artificial Potential Field (APF) combined with Virtual Obstacles (VOs) to handle local minima. The 3D potential map generated guides robots from their initial positions to the goal, avoiding collisions. Simulations in MATLAB and V-REP demonstrate the effectiveness of this approach in real-time applications.

III. METHODS

We model multiple AGVs that follow a thick, dark, self-intersecting line. Each AGV is equipped with line sensors and ultrasound detectors to detect the line and avoid collisions with other AGVs. The intersections are equipped with sensors to detect incoming vehicles and traffic lights to signal those vehicles to stop or move. The entire system can be modeled as a hybrid timed system.

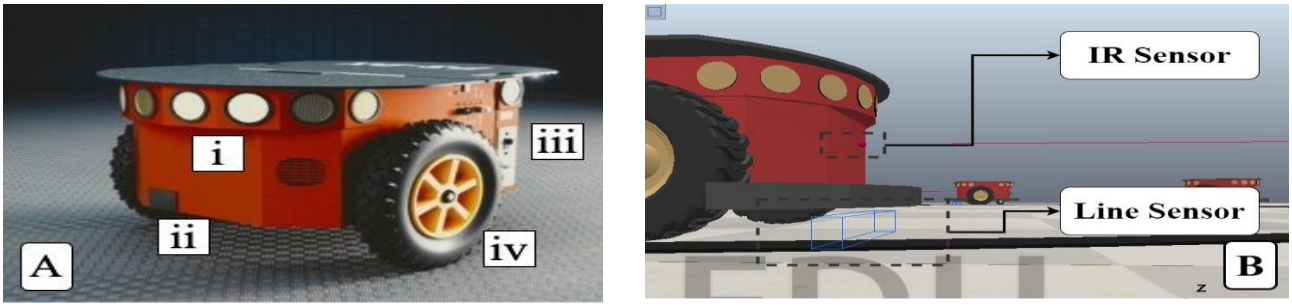

Figure 1 A: Pioneer 3-DX robot (i: Side lights, ii: IR sensor, iii: controls for power, iv: two differential drive wheels; B: Sensors on the Pioneer 3DX Robot

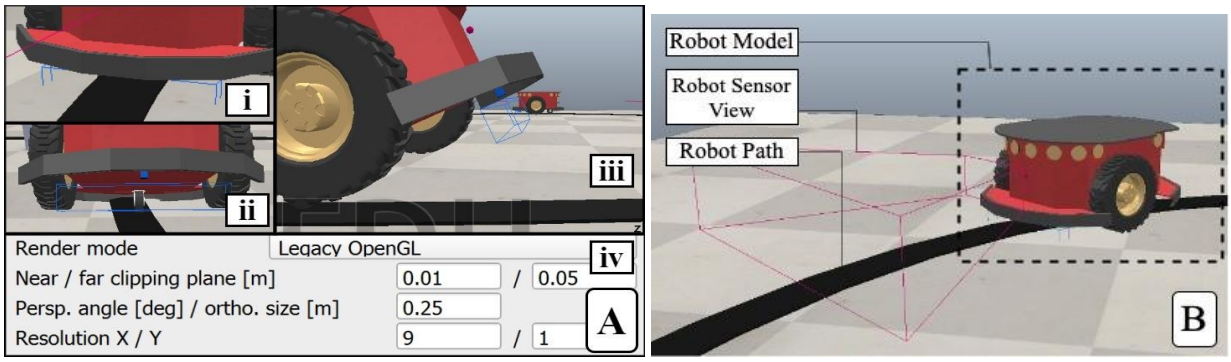

Figure 2 Robot Line Sensor - A: (i) Top View; (ii): Bottom View; (iii): Side View; (iv): Line Sensor Settings in VREP; B: Robot Proximity

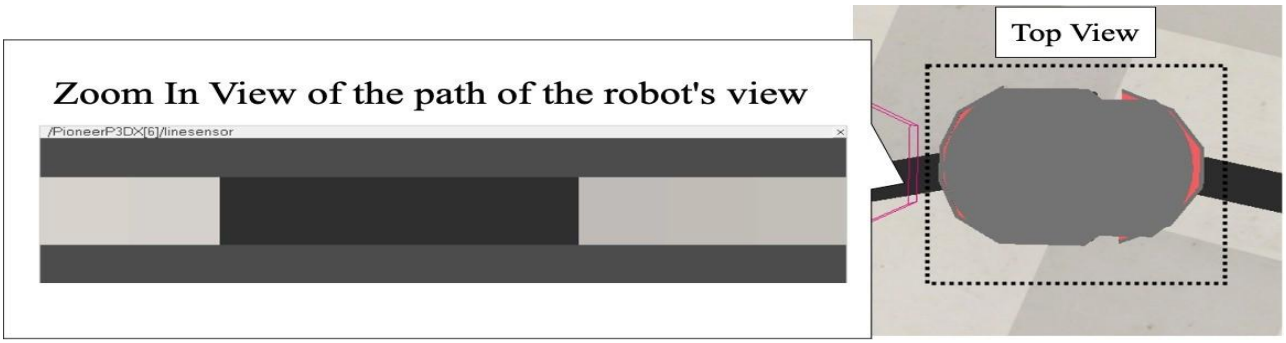

Figure 3 What the Robot sees

A. Vehicle Model

The Pioneer 3-DX robot, used for our AGV model, is a small, lightweight, two-wheel differential drive robot, ideal for indoor labs or classrooms. The simulated vehicle model includes a downward-facing line sensor, typically around 20-50 mm wide and 5-20 mm deep, mounted 5-15 mm above the ground for detecting lines, and a forward-facing ultrasound sensor [30]. However, the simulation utilizes an IR sensor as shown in Figure 1. [31]

B. Path Tracking

The line sensor is modeled as a 1x9 image sensor that captures a color image of the ground below. The width of the path is 0.05m or 5 cm. And the length of path is ~25m. The center of the black stripe in this image is identified, and the output of this method is the deviation of this center from the image center as shown in Figure 3. The line sensor also detects traffic signals on the floor.

C. Path Following

1) Path following Algorithm:

The script is programmed with LUA in Coppealiasim VREP. [32] Algorithm 1 shows a robot in a simulation environment, using motors, a line sensor, and a proximity sensor. During initialization, it retrieves necessary object handles and sets initial parameters for line following and obstacle detection. The actuation phase adjusts motor speeds based on whether the robot needs to stop or avoid an obstacle, setting target velocities accordingly. The sensing phase processes image data from the line sensor to calculate line-following errors and checks the proximity sensor for obstacles.

The parameters and their initial numerical values are as follows: `line_err` is set to 0, `base_spd` is 5, `obs` is initialized to 0, `lw` or line width is 9 cm for sensing, with `obs_dist` set to 10 cm and `obs_thresh` to 0.2. The `stop` and `go` flags are both initialized to 0.

Algorithm 1 Robot Control Algorithm

Initialization:
Set obsDist = 0.10m obsThrsh = 0.2m
Retrieve motor and sensor handles
Initialize variables for line following and obstacle detection (leftMotor, rightMotor, lineSensor, proximity, lineError, baseSpd, obsDist, obsThresh)
Actuation:
Set base speed
if stop flag is set **then**
Set base speed to 0
else if obstacle detected **then**
Adjust base speed based on obstacle distance -> baseSpd = baseSpd*2.5*(obsDist-obsThresh)
end if
Set motor speeds based on base speed and line error
Sensing (Line Sensor):
Retrieve and unpack image from line sensor
Initialize sums and weights for line detection
for each pixel group in the image **do**
Calculate negative value based on pixel colors
Update red and green sums according to Eqn. 5, 6
Update weighted sums
end for
Calculate center of line and line error
if red sum exceeds threshold, **then**
Set stop flag
else
Clear stop flag
end if
if green sum exceeds threshold, **then**
Set go flag
else
Clear go flag
end if
Read proximity sensor
while object range < 0.50m -> object detection
Continue

Algorithm 2 Traffic Intersection Control Algorithm

Initialization:
Get references to traffic lights and sensors
Initialize variables: xw, yw, ins, x1, y1, ss to 0 (xw, yx – robot counter, x1,y1 – traffic light state, ss- state variable for switching lights, ins – number of robot in the system)
Sensing (Intersection):
Read sensor values (xa, ya, xe, ye, xex, yex) (Approach, enter and exit blocks)
Update previous sensor values
Process Events:
Approach:
if vehicle starts approaching (sensor values change) **then**
update xw and yw
end if
Enter:
if vehicle enters (sensor values change) **then**
update xw, yw, and ins
end if
Exit:
if vehicle exits (sensor values change) **then**
update ins
end if
Compute Signal STATE:
if ins > 0 **then**
set both lights to red
else if both xw and yw > 0 **then**
Toggle signal STATE ss
Set light colors based on ss
else if xw > 0 or yw > 0 **then**
Set light colors accordingly
end if
Switch Lights:
Set X light and Y light colors based on computed signals

The robot stops if a red line is detected and is ready to go if a green line is detected, with obstacle distance affecting speed adjustments. The cleanup phase is reserved for any required actions when the simulation ends.

Algorithm 2 begins with initializing references to traffic lights and sensors, followed by reading sensor values to detect vehicle movements. The algorithm processes various vehicle events such as approaching, entering, and exiting the intersection, updating its state accordingly. It then computes the appropriate signal state based on real-time data, ensuring that traffic lights are set to red when necessary and toggling signal states to manage traffic flow. Finally, it sets the traffic light colors based on the computed signals, ensuring smooth and safe passage through the intersection.

In the context of the robot's line-following logic, the variable lw represents a critical parameter that defines the width of the line the robot is expected to follow. This parameter is then used to compute several derived variables that are essential for error calculation and decision-making in the robot's control system.

1. Line Width (lw)

The variable lw is defined as a constant integer value representing the base width of the line in pixels as perceived by the vision sensor.

2. Derived Parameters

Several derived parameters are calculated based on lw:

$$\mathrm{lw1} = \frac{3 \times \mathrm{l_w} + 1}{2} \quad (1)$$

$$\mathrm{lw2} = \frac{3 \times \mathrm{l_w} - 1}{2} \quad (2)$$

$$\mathrm{lw3} = 0.67 \times 3 \times \mathrm{l_w} \times 255 \quad (3)$$

- **lw1**: Represents the center of the line's width, scaled by a factor of 3/2 with a small adjustment. This value is used to normalize the position of the line error.
- **lw2**: Represents a similarly scaled but slightly reduced width, used as the denominator in the normalization of the line error calculation.
- **lw3**: Represents a threshold for the summed RGB values obtained from the vision sensor, scaled by a factor of 0.67 x 3 x lw. The factor 255 is used to scale

the values to match the maximum possible RGB value in 8-bit color depth (since each color channel can have a maximum value of 255).

3. **Error Calculation (line_err)**
   The error in line following, line_err, is calculated using the weighted sum of pixel values captured by the vision sensor. This error represents the deviation of the robot from the center of the line.

$$\text{cent} = \frac{\sum_{i=1}^{3\times 1w} (i \times \text{neg})}{1 \times 10^{-6} + \sum_{i=1}^{3\times 1w} \text{neg}} \quad (4)$$

$$\text{line\_err} = \frac{\text{cent} - \text{lw1}}{\text{lw2}} \quad (5)$$

- **cent**: This is the weighted center of the line detected by the vision sensor, computed as a ratio of the weighted sum of pixel indices to the total negative sum (neg). A small value of 10^-6 is added to avoid division by zero.
- **line_err**: Represents the normalized deviation of the line's perceived center from the robot's expected center (lw1). This error is then used to adjust the motor velocities, guiding the robot back toward the line's center.

4. **Stop and Go Conditions**
   - The robot exceeds the sum of the red and green pixel values exceeds lw3. If the red sum is greater than lw3, the robot stops (stop = 1). If the green sum is greater than lw3, the robot initiates movement (go = 1). See Equation 10 and 11 how sum of red and green pixel is calculated.

2) Error Handling:

The deviation error from the line sensor is fed into a Proportional-Integral-Derivative (PID) controller to generate a control signal. This control signal adjusts the speeds of the two wheels, causing the vehicle to turn and align itself with the line. The line error, *line_err*, indicates how far the robot is from the center of the line it should follow.

It is calculated as Equation 6. The following equation is generalized from Equation 4 and 5:

$$\text{line\_err} = \frac{\text{centroid} - \text{midpoint}}{\text{normalizer}} \quad (6)$$

where *centroid* is the position of the line detected by the sensor, *midpoint* is a reference value representing the center position, and *normalizer* is a factor used to scale the error.

The speed of the robot's motors is adjusted based on the line error and whether the robot should stop or avoid obstacles. The motor speeds are given by Equation 7 and 8:

$$\text{leftMotorSpeed} = \text{baseSpeed} \times (1 + \text{line\_err}) \quad (7)$$

$$\text{rightMotorSpeed} = \text{baseSpeed} \times (1 - \text{line\_err}) \quad (8)$$

where *baseSpeed* is the default speed of the motors and *line_err* is the calculated error from the line.

D. Obstacle Avoidance & Traffic Detection

The ultrasonic/radar/lidar sensor in the front of the AGV looks for obstacles in front of it. If an obstacle is detected, the AGV is put to stop a specified distance behind the obstacle. The following Equation 9 is followed where base speed of robot is adjusted based on object distance and object threshold

$$baseSpd = baseSpd \times 2.5 \times (obsDist - obsThresh) \quad (9)$$

where *baseSpd* is the current speed of the robot's motors, *obsDist* is the distance to the nearest obstacle detected by the proximity sensor, and *obsThresh* is the threshold distance below which the robot starts slowing down.

Traffic lights are installed on the floor as coloured lights. The line sensor image is used to detect red/green lights on the floor. The vehicle keeps going if it detects a green light. The vehicle stops on detecting a red light. The traffic light system consists of 2 proximity sensors that detect incoming vehicles, and 2 colored stripes on the floor.

The robot detects whether it should stop or go based on the colors detected by the sensor. This is done by summing the color values as Equation 10 and 11:

$$\text{red}_{\text{sum}} = \sum_{i=1,3}^{3\cdot ww} (510 + \text{img}[i] - \text{img}[i+1] - \text{img}[i+2]) \quad (10)$$

$$\text{green}_{\text{sum}} = \sum_{i=1,3}^{3\cdot lw} (510 - \text{img}[i] + \text{img}[i+1] - \text{img}[i+2]) \quad (11)$$

where *red_sum* represents the sum of red pixel values, and *green_sum* represents the sum of green pixel values. If *red_sum* is high, the robot stops; if *green_sum* is high, the robot continues moving.

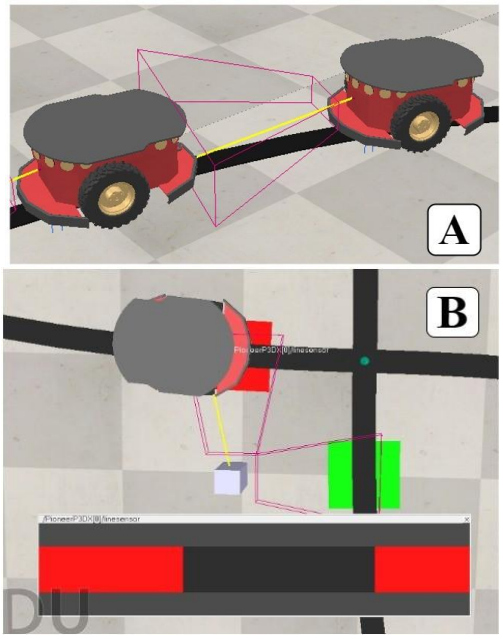

Figure 4 Obstacle Avoidance and Traffic Management: A: Robot detecting robot; B: Robot detecting Traffic line

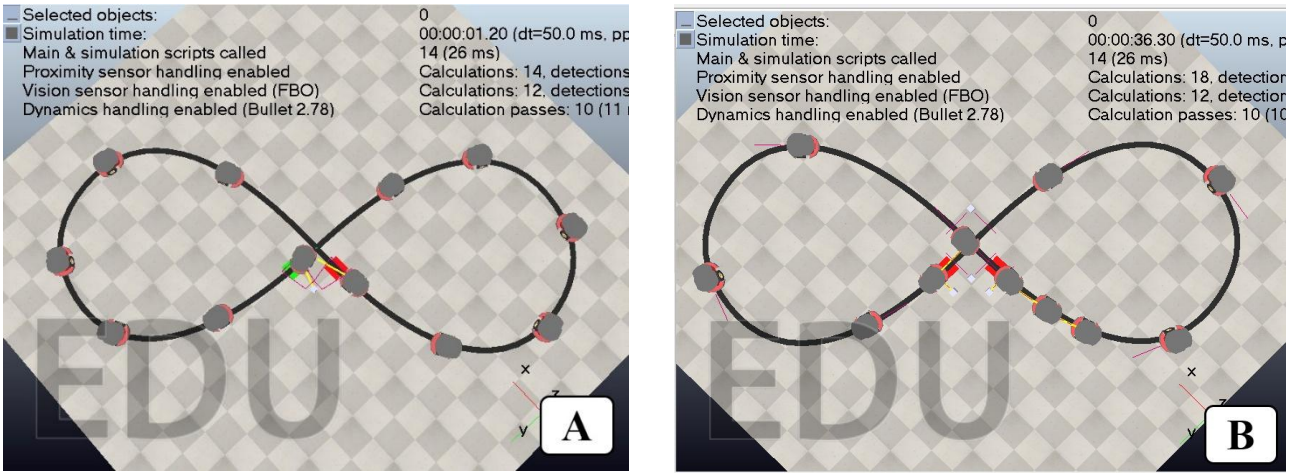

Figure 5 Robot simulation demo in Coppeliasim VREP: A - E1 - Entrance Block only; B - E2 - Entrance, Approach, Exit Block

As shown in Figure 4 the traffic light system consists of 2 proximity sensors that detect incoming vehicles and 2 colored stripes on the floor. These stripes can change color to inform incoming vehicles to stop or go. In this system, it is required that at the same time, there should not be 2 vehicles at the intersection, causing a collision. Both sides should also be able to go in a reasonable amount of time (no starvation).

IV. Results & Discussion

In the context of a traffic management system, the following logic is implemented to optimize the flow of vehicles at an intersection:

1) **No Vehicles Waiting:**
   - No change in the traffic light STATE is required if no vehicles are waiting. The traffic lights remain in their current configuration.
2) **One Vehicle Waiting:**
   - If there is a vehicle waiting on one side, the traffic light on that side turns green, while the light on the opposite side turns red. This ensures that the waiting vehicle can proceed without unnecessary delay.
3) **Vehicles Waiting on Both Sides:**
   - When vehicles are waiting on both sides, the system randomly selects one side to turn green, and the other side turns red. This approach prevents starvation, ensuring that neither side is perpetually waiting while the other side continuously receives the green light.
   -

However, the system can cause the traffic lights to "chatter," meaning they switch STATEs frequently when vehicles are present on both sides. To mitigate this issue, a timed automaton is introduced. The timed automaton enforces a fixed interval during which the traffic lights remain in their current STATE before any changes are made. This approach prevents rapid oscillation of the traffic lights, promoting a smoother and more predictable flow of traffic.

As shown in Figure 5A, the model avoids congestion using sensors using intelligent scheduling. It showcases the E1 scenario where there is only one sensor block before the intersection i.e. the entrance block. By implementing a timed automaton, the traffic management system achieves a balance between responsiveness to vehicle presence and stability in traffic light operation, thereby enhancing overall traffic efficiency and reducing potential congestion at the intersection.

As shown in Figure 5B, there is an entrance block to check robots coming in the intersection, an approach block to sense they are approaching and an exit block to indicate the robots are exiting the system.

A. Results

The path chosen for the simulation was a spiral in nature, and the total number of AGVs was 12, split evenly with 6 on each side of the spiral pathway. The simulation was conducted in two distinct scenarios, as illustrated in Figure 5 and Figure 6.

1) Test Scene E1 – Single Entrance at Intersection

In the first scenario (E1), the simulation incorporated an entrance block designed to detect vehicles entering the spiral pathway. The purpose of this setup was to observe how AGVs manage congestion with a single detection point. Figure 6 (Blue) presents the simulation results for this case, highlighting the relationship between vehicle speed and congestion at the intersection.

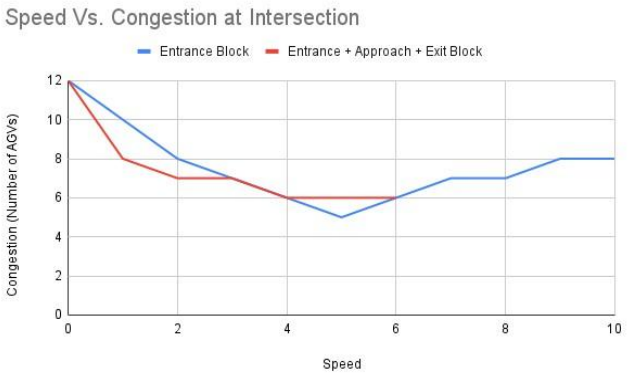

Figure 6 AGVs simulation results - Blue E1 - Entrance Block only scene; Red E2 - Entrance, Approach, Exit Block scene

2) Test Scene E2 – Entrance, Approach and Exit block at Intersection

In the second scenario (E2), the simulation was expanded to include an entrance block, an approach block, and an exit block. This setup provided a more complex traffic management system, aiming to evaluate how additional detection points influence traffic flow and congestion dynamics. Figure 6 (Red) illustrates the results for this case, offering a comparative analysis against the E1 scenario. The presence of additional blocks allowed for a more comprehensive understanding of congestion behavior as vehicles moved through the spiral pathway.

3) Analysis of Simulation Results

Table I complements the graphical data by detailing congestion levels associated with different vehicle speeds, ranging from 1 to 10 units or 0 to 97 cm/s. In both scenarios, the speeds on the left side of the spiral increased incrementally with each test, while the speeds on the right side remained constant at 48 cm/s. The logic to calculate the speed of the object in the simulation is straightforward and involves a few key steps. First, the current position of the object is obtained using `sim.getObjectPosition`, which returns the coordinates (x, y, z) in the simulation space. The time at which this position is recorded is also captured using `sim.getSimulationTime`.

To calculate the speed, the difference in position between the current and the previous frame is computed for each coordinate axis (x, y, and z). This difference represents the displacement in each direction. The total distance traveled by the object is then calculated using the Euclidean distance formula in Equation 12:

$$\text{distance} = \sqrt{(dx)^2 + (dy)^2 + (dz)^2} \qquad (12)$$

where `dx`, `dy`, and `dz` are the differences in the x, y, and z coordinates, respectively.

The time difference (`dt`) between the current and previous frames is calculated, and the speed is then derived by dividing the distance by the time difference. The speed is expressed in centimeters per second (cm/s). Finally, the previous position and time are updated to the current values to be used in the next iteration of the loop, allowing for continuous speed calculation. The calculated speed can be printed or stored for further analysis or debugging purposes. Notable observations include higher congestion at lower speeds and a significant increase in congestion with higher speeds, particularly in the entrance block (E1).

Congestion was high at slower speeds but reduced as the motor speed increased to around 48–68 cm/s. However, when the speeds increased further, up to 97 cm/s, while keeping the right spiral's speed constant, the congestion also increased.

Table 1 Simulation results of robot speed vs congestion. E1 - Entrance Block only scene; E2 - Entrance, Approach, Exit Block scene

Robot Speed (cm/s)		**Max Congestion**		**Remarks**	
Left Circle	**Right Circle**	**E1**	**E2**	**Speed**	**Congestion**
0	48	12	12	None	Block
9	48	10	8	Slow	High
19	48	8	7	Slow	High
29	48	7	7	Moderate	Medium
38	48	6	6	Moderate	Medium
48	48	5	6	Normal	Medium
58	48	6	6	Normal	Medium
68	48	7	NA	Above average	Rising
77	48	7	NA	Increasing	E1 increasing, E2 Fails
87	48	8	NA	High	E1 increasing, E2 Fails
97	48	8	NA	High	E1 increasing, E2 Fails

However, the same cannot be said for E2 as at high speeds the system fails even though it has 3 sensor blocks. The reason for this is robots with high speed have a harder time halting speed at each sensor block. This indicates that maintaining a moderate and constant speed limit around intersections is crucial for optimal traffic flow.

B. Limitations

Estimating when a vehicle has entered and/or exited the intersection is challenging due to the non-zero length of vehicles, which cannot be treated as points. This complexity is further compounded by the difficulty in identifying the number and types of sensors required for accurate detection and management. Additionally, it has not been formally proven that the proposed system avoids crashes, indicating potential safety concerns that need to be addressed.

V. CONCLUSIONS

This paper addresses the critical challenge of path tracking and traffic management for AGVs in complex urban environments. The proposed approach integrates advanced control mechanisms and sensors to mitigate congestion and enhance operational efficiency at traffic control intersections. Simulation results in Coppeliasim VREP demonstrate that AGVs maintained normal to moderate speeds in high congestion scenarios, reduces maximum congestion 20% ensuring continuous flow. By leveraging IR-based sensors and a novel path-tracking technique, the system aims to reduce waiting times, avoid collisions, and optimize the flow of AGVs.

The simulation results demonstrate the effectiveness of the proposed method in managing vehicle traffic at intersections,

as shown in the Coppeliasim VREP environment. The timed automaton approach successfully minimizes traffic light oscillation, ensuring a more stable and predictable flow of vehicles. This approach helps balance responsiveness to vehicle presence with the need for stable traffic light operation, thus enhancing overall system efficiency.

However, there are limitations to the current system. Challenges remain in accurately estimating vehicle entry and exit times at intersections, especially given the nonzero length of vehicles. In high-speed scenarios congestion increases. Additionally, while the system is designed to reduce collisions and manage traffic flow effectively, formal proof of crash avoidance is still required to address potential safety concerns.

Future work should focus on refining sensor accuracy, improving the robustness of the traffic management algorithms, and validating the system's safety through extensive testing and formal proofs. Overall, the proposed system provides a promising solution for managing AGV traffic in urban environments, contributing to the advancement of autonomous vehicle technology and its application in smart transportation systems.

REFERENCES

[1] C. Wang and J. Mao, "Summary of agv path planning," in 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE), pp. 332–335, IEEE, 2019.

[2] I. Kubasakova, J. Kubanova, D. Benco, and D. Kadlecová, "Implementation of automated guided vehicles for the automa- tion of selected processes and elimination of collisions between handling equipment and humans in the warehouse," Sensors, vol. 24, no. 3, p. 1029, 2024.

[3] J. Zhang, X. Yang, W. Wang, J. Guan, L. Ding, and V. C. Lee, "Automated guided vehicles and autonomous mobile robots for recognition and tracking in civil engineering," Automation in Construction, vol. 146, p. 104699, 2023.

[4] Y. Zhang, F. Wang, F. Fu, and Z. Su, "Multi-agv path planning for indoor factory by using prioritized planning and improved ant algorithm.," Journal of Engineering & Technological Sciences, vol. 50, no. 4, 2018.

[5] S. Arora and A. K. Mittal, "A priority based conflict resolution approach for automated guided vehicles," in 2006 American Control Conference, pp. 6–pp, IEEE, 2006.

[6] A. J. Moshayedi, G. Xu, L. Liao, and A. Kolahdooz, "Gentle survey on mir industrial service robots: review & design," J. Mod. Process. Manuf. Prod, vol. 10, no. 1, pp. 31–50, 2021.

[7] A. J. Moshayedi, J. Li, and L. Liao, "Simulation study and pid tune of automated guided vehicles (agv)," in 2021 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applica- tions (CIVEMSA), pp. 1–7, 2021.

[8] A. J. Moshayedi, K. S. Reza, A. S. Khan, and A. Nawaz, "Integrating virtual reality and robotic operation system (ros) for agv navigation," EAI Endorsed Transactions on AI and Robotics, vol. 2, 2023.

[9] A. J. Moshayedi, A. Sohail Khan, J. Hu, A. Nawaz, and J. Zhu, "E-nose-driven advancements in ammonia gas detection: a com- prehensive review from traditional to cutting-edge systems in indoor to outdoor agriculture," Sustainability, vol. 15, no. 15, p. 11601, 2023.

[10] A. J. Moshayedi, A. S. Roy, L. Liao, A. S. Khan, A. Kolahdooz, and A. Eftekhari, "Design and development of foodiebot robot: From simulation to design," IEEE Access, 2024.

[11] G. Xu, A. S. Khan, A. J. Moshayedi, X. Zhang, and Y. Shuxin, "The object detection, perspective and obstacles in robotic: a review," EAI Endorsed Transactions on AI and Robotics, vol. 1, no. 1, 2022.

[12] A. J. Moshayedi, A. S. Khan, Y. Yang, J. Hu, and A. Kolahdooz, "Robots in agriculture: Revolutionizing farming practices," EAI Endorsed Transactions on AI and Robotics, vol. 3, 2024.

[13] A. J. Moshayedi, J. Li, N. Sina, X. Chen, L. Liao, M. Gheisari, and X. Xie, "Simulation and validation of optimized pid con- troller in agv (automated guided vehicles) model using pso and bas algorithms," Computational Intelligence and Neuroscience, vol. 2022, no. 1, p. 7799654, 2022.

[14] A. J. Moshayedi, Y. Xie, M. Sharifdoust, and A. S. Khan, "Evaluating omni robot navigation with slam in coppeliasim: Hemangiomas and nonhomogeneous paths," Journal of Robotics Research (JRR), vol. 1, no. 1, pp. 7–14, 2024.

[15] A. J. Moshayedi, A. S. Roy, and L. Liao, "Pid tuning method on agv (automated guided vehicle) industrial robot," Journal of Simulation and Analysis of Novel Technologies in Mechanical Engineering, vol. 12, no. 4, pp. 53–66, 2019.

[16] A. J. Moshayedi, S. M. Zanjani, D. Xu, X. Chen, G. Wang, and S. Yang, "Fusion based agv robot navigation solution compar- ative analysis and vrep simulation," in 2022 8th Iranian Con- ference on Signal Processing and Intelligent Systems (ICSPIS), pp. 1–11, IEEE, 2022.

[17] X. Deng, B. Zhou, X. Sun, H. Yang, and L. Chen, "A method for reliability detection of automated guided vehicle based on timed automata," Systems Science & Control Engineering, vol. 9, no. 1, pp. 570–579, 2021.

[18] L. Yue and H. Fan, "Dynamic scheduling and path planning of automated guided vehicles in automatic container termi- nal," IEEE/CAA Journal of Automatica Sinica, vol. 9, no. 11, pp. 2005–2019, 2022.

[19] J. Feng, Y. Yang, H. Zhang, S. Sun, and B. Xu, "Path planning and trajectory tracking for autonomous obstacle avoidance in automated guided vehicles at automated terminals," Axioms, vol. 13, no. 1, p. 27, 2023.

[20] J. Hu, B. Yu, B. Zhu, and W. Wu, "A congestion-predicting path planning algorithm for narrow lanes in automated ware- houses," in 2024 36th Chinese Control and Decision Conference (CCDC), pp. 1515–1520, IEEE, 2024.

[21] Ergin[1], Simge Güçlükol, and Mahmut Ali Gökçe. "Multi Objective Optimization for Intelligent Scheduling and Routing of Automated Guided Vehicle (AGV) Assisted Order Picking." *Intelligent and Fuzzy Systems:*

Intelligent Industrial Informatics and Efficient Networks Proceedings of the INFUS 2024 Conference, Volume 2. Springer Nature.

[22] K. Lee and S. Park, "Geometric zone-control algorithm for collision and deadlock avoidance in agv system," IEEE Access, vol. 11, pp. 131289–131301, 2023.

[23] Q. Wang and C. Wang, "Exploration of port intelligent agv path tracking based on vision," Journal of Intelligent & Fuzzy Systems, vol. 38, no. 2, pp. 1281–1285, 2020.

[24] P. Verma, J. M. Olm, and R. Suárez, "Traffic management of multi-agv systems by improved dynamic resource reservation," IEEE access, 2024.

[25] Q. Yang, Y. Lian, Y. Liu, W. Xie, and Y. Yang, "Multi-agv tracking system based on global vision and apriltag in smartwarehouse," Journal of Intelligent & Robotic Systems, vol. 104, no. 3, p. 42, 2022.

[26] S. Zhou, G. Cheng, Q. Meng, H. Lin, Z. Du, and F. Wang, "Development of multi-sensor information fusion and agv nav- igation system," in 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), vol. 1, pp. 2043–2046, IEEE, 2020.

[27] Z. Han, D. Wang, F. Liu, and Z. Zhao, "Multi-agv path planning with double-path constraints by using an improved genetic algorithm," PloS one, vol. 12, no. 7, p. e0181747, 2017.

[28] C. Liu, J. Tan, H. Zhao, Y. Li, and X. Bai, "Path planning and intelligent scheduling of multi-agv systems in workshop," in 2017 36th Chinese Control Conference (CCC), pp. 2735–2739, IEEE, 2017.

[29] A. M. Hassan, C. M. Elias, O. M. Shehata, and E. I. Morgan, "A global integrated artificial potential field/virtual obstacles path planning algorithm for multi-robot system applications," Int. Research J. of Eng. and Technology, vol. 4, no. 9, pp. 1198– 1204, 2017.

[30] C.-H. Chen, C.-J. Lin, J. Shiou-Yun, H.-Y. Lin, and C.-Y. Yu, "Using ultrasonic sensors and a knowledge-based neural fuzzy controller for mobile robot navigation control," Electronics, vol. 10, p. 466, 02 2021.

[31] S. I. Martínez, J. A. C. Rocha, J. L. Menchaca, M. G. T. Berrones, J. G. Obando, J. P. Cobos, and E. C. Rocha, "An autonomous navigation methodology for a pioneer 3dx robot," Computer Technology and Application, vol. 5, no. 2, 2014.

[32] Rohmer, Eric, Surya PN Singh, and Marc Freese. "V-REP: A versatile and scalable robot simulation framework." *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013