IRSTI 28.19.23 DOI: https://doi.org/10.62687/STJ.1.2.2026.17

COMPARATIVE ANALYSIS OF 3D MAPPING ALGORITHMS FOR ENVIRONMENTAL EXPLORATION ON GRAPH BASIS

[1]A.A. Zhulumbetov*, **[1]T.K. Zhukabayeva**
[1]Astana IT University, Astana, Kazakhstan
*e-mail: zhulumbetov.ar@gmail.com

A.A. Zhulumbetov – master's student, Astana IT University, Astana, Kazakhstan, e-mail: zhulumbetov.ar@gmail.com, https://orcid.org/0009-0008-7588-5559
T.K. Zhukabayeva – professor, Astana IT University, Astana, Kazakhstan, e-mail: tamara.kokenovna@gmail.com, https://orcid.org/0000-0001-6345-5211

Abstract. In a modern topology many robots already use the existed map structures, but developing an autonomous system in a previously unknown environment with minimal data and a limited field of view has not yet been definitively explored. Many features of the cities or some area create a closed route, which robots may not leave without human intervention. This paper compares the proposed hybrid graph-construction system with existing algorithms, RRT* and D*Lite, for navigation in an unknown environment; deterministic paths are used in open terrain for obstacle avoidance.

The developed algorithm will create a topographic graph that creates nodes for the future graph, which will then create loops. Once the loop detected the algorithm will use these nodes to exit the loop for further exploration of the terrain.

The goal of this paper is to analyze how the graph construction algorithm performs in a dynamic environment, compared to RRT* and D*Lite, where objects are static and dynamic. To achieve the goal, the following objectives will be performed: constructing a 2D map with random objects and destinations, training neural agents to detect possible nodes and exit them, and constructing an optimal path from point to point using minimal computation in a sufficiently open area.

The three methods were simultaneously simulated in 40 different situations with different initial parameters. The hybrid path construction method achieved the goal in half of all tests, where on average they approached the goal by 27.7%. The highest result was shown by RRT* at 37.3%, only it took an average of 1409 nodes to achieve the goal, which is twice as much as the proposed method in the work. D*Lite completed the training with a score of 3% of the distance traveled, as closed loops were created with complex calculations and heavy traffic.

Keywords: 3D environment mapping, frontier exploration, cycle detection, RRT*, D* Lite, reinforcement learning, dynamic obstacles, urban navigation.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ 3D-КАРТИРОВАНИЯ ДЛЯ ИССЛЕДОВАНИЯ ОКРУЖАЮЩЕЙ СРЕДЫ НА ОСНОВЕ ГРАФОВ

[1]А.А. Жулумбетов*, [1]Т.К. Жукабаева
[1]Астана IT Университет, Астана, Казахстан
*e-mail: zhulumbetov.ar@gmail.com

А.А. Жулумбетов – магистрант, Астана IT Университет, Астана, Казахстан, e-mail: zhulumbetov.ar@gmail.com, https://orcid.org/0009-0008-7588-5559
Т.К. Жукабаева – профессор, Астана IT Университет, Астана, Казахстан, e-mail: tamara.kokenovna@gmail.com, https://orcid.org/0000-0001-6345-5211

Аннотация. В современной топологии многие навигационные роботы часто используют ранее изученную местность и используют построенные графы, но разработка автономной системы в ранее неизвестной среде при минимальных данных и ограниченном поле

зрении еще не были однозначно разработаны. Многие структуры городов или комплексные местности создают замкнутые циклы, из-за которых автономные роботы не могут выйти без стороннего вмешательства. Данная работа рассматривает сравнение со существующими алгоритмами RRT* и D* Lite с построением графа в неизвестной среде с интеграцией обхода препятствий с использованием детерминированных путей при открытости местности.

Созданный алгоритм в работе строит топографический граф, который создает узлы для будущего графа, которые образуют циклы. Когда цикл построен, через узлы использован алгоритм для выхода из цикла для дальнейшего исследования местности.

Целью работы является анализ как себя проявляет алгоритм построения графа в динамической среде в сравнении с RRT* и D* Lite, где объекты статичные и динамичные. Для достижения цели были выполнены следующие задачи: построение 2D карты с случайными объектами и пунктами назначения, обучение нейронных агентов для обнаружения возможных узлов и выхода из них, построение оптимального пути от точки до точки, используя минимум вычислений при достаточно открытой местности.

Три метода были простимулированы одновременно на 40 разных ситуациях с разными изначальными параметрами. Гибридный метод построения пути достиг цели в половине всех тестов, где в среднем они приблизились к цели на 27.7%. Самый высокий результат показал RRT* в 37.3%, только для достижения цели потребовалось в среднем 1409 узлов, что вдвое больше, чем предложенный метод в работе. D* Lite завершил обучение с результатом 3% пройденного пути, так как при сложных вычислениях и под интенсивным движением преград были созданы замкнутые циклы.

Ключевые слова: 3D-картирование окружающей среды, исследование новых территорий, обнаружение циклов, RRT*, D* Lite, обучение с подкреплением, динамические препятствия, городская навигация.

ҚОРШАҒАН ОРТАНЫ ЗЕРТТЕУГЕ АРНАЛҒАН ГРАФҚА НЕГІЗДЕЛГЕН 3D КАРТАЛАУ АЛГОРИТМДЕРІНІҢ САЛЫСТЫРМА ТАЛДАУЫ

[1]А.А. Жулумбетов*, [1]Т.К. Жукабаева
[1]Астана IT Университеті, Астана, Қазақстан
*e-mail: zhulumbetov.ar@gmail.com

А.А. Жулумбетов – магистрант, Астана IT Университеті, Астана, Қазақстан, e-mail: zhulumbetov.ar@gmail.com, https://orcid.org/0009-0008-7588-5559

Т.К. Жукабаева – профессор, Астана IT Университеті, Астана, Қазақстан, e-mail: tamara.kokenovna@gmail.com, https://orcid.org/0000-0001-6345-5211

Аңдатпа. Қазіргі топологияда көптеген навигациялық роботтар бұрын зерттелген аумақты жиі пайдаланады және бар графиктерді пайдаланады. Дегенмен, бұрын белгісіз ортада минималды деректермен және шектеулі көру өрісімен автономды жүйені әзірлеу әлі нақты қарастырылған жоқ. Көптеген қалалық құрылымдар немесе күрделі жер бедері автономды роботтардың сыртқы араласусыз қашып кетуіне кедергі келтіретін тұйық циклдар жасайды. Бұл мақалада оны белгісіз ортада граф құру үшін қолданыстағы RRT* және D* Lite алгоритмдерімен салыстырады, ашық жердегі детерминистік жолдарды пайдаланып кедергілерден аулақ болуды біріктіреді.

Әзірленген алгоритм граф үшін түйіндер жасайтын топографиялық график жасайды, ол содан кейін циклдар жасайды. Цикл құрылғаннан кейін, алгоритм жер бедерін одан әрі зерттеу үшін циклдан шығу үшін осы түйіндерді пайдаланады.

Бұл мақаланың мақсаты - граф құру алгоритмінің динамикалық ортада қалай жұмыс істейтінін талдау, мұнда нысандар статикалық және динамикалық болып табылатын RRT* және D* Lite-пен салыстырғанда. Мақсатқа жету үшін келесі міндеттер орындалады: кездейсоқ нысандар мен бағыттармен 2D картасын құру, нейрондық агенттерді мүмкін түйіндерді

анықтауға және олардан шығуға үйрету және жеткілікті ашық аймақта минималды есептеуді қолдана отырып, нүктеден нүктеге оңтайлы жол салу.

Үш алгоритмнің барлығы кездейсоқ граф жасалған 40 жасалған дәуірде модельденді. Ұсынылған әдіс мақсатқа 40 дәуірдің 20-сында қол жеткізді, мақсатқа жету жолының орташа аяқталу деңгейі 27,7% құрады. RRT* ең жоғары нәтижеге, яғни 37,3% жетті, мақсатқа жету үшін орташа есеппен 1409 түйін қажет болды, бұл мақалада ұсынылған әдістен екі есе көп. D* Lite күрделі есептеулер мен кедергілердің қарқынды қозғалысы тұйықталған циклдарды тудырғандықтан, мақсатқа жету жолының аяқталу деңгейі 3% болды.

Түйін сөздер: 3D қоршаған ортаны картаға түсіру, жаңа аумақты зерттеу, циклды анықтау, RRT*, D* Lite, күшейту бойынша оқыту, динамикалық кедергілер, қалалық навигация.

Introduction. Autonomous navigational system plays an important role in the creation of the safe and effective route in different topological situations. Due to the different city structures with various densities and constantly changing dynamic objects the achievement of high efficiency in the route creation became a challenging concept.

Traditional path planning like Djikstra, A* mostly work on the predefined map structure. Another algorithm RRT uses the technique that draws a probabilistic tree path, which partially works in the unknown environmental. However, they have a limitation in a dynamic structure, where objects might block the passage for short period of time. Experimental results in the paper A Path-Planning Performance Comparison of RRT*-AB with MEA* in a 2-Dimensional Environment (Noreen et al., 2019) demonstrate effective use of the RRT and RRT* in the route creation in complex non-dynamic scenarios. However, the results show that RRT requires over 140 seconds to construct the path, while A* requires 12 seconds, with fewer nodes required. This time complexity is connected to the global initial planning of the RRT, which shows its disadvantage if the passage can be blocked for the short period of time, if such situation appeared the new global replanning will be constructed. Therefore, paper proposes a new method of the hybrid frontier-based exploration, which creates a global path and has the reactive response to the dynamic obstacles and attempts reconstruct the passage around the obstacle.

One of the main problems in such situations the creation of the closed loops, where navigational agents can be locked in a dead-end, without knowing the global structure. This is particularly true in complex urban landscapes, where buildings and other obstacles form topologically closed areas.

This paper proposes an approach based on cycle-aware graph construction, implemented within the Cycle-Aware Frontier Exploration method. The proposed method utilizes a multi-agent system in which agents create graph nodes in an unknown environment and form closed structures that enable more efficient detection of object boundaries and organization of further spatial exploration (Waga et al., 2025). Unlike traditional methods, the proposed approach combines graph construction with elements of reinforcement learning, enabling adaptation to dynamic changes in the environment. Agents exchange information about the locations of nodes without transmitting a complete picture of the surrounding space, which reduces the computational load and makes the system more scalable (Zhu et al., 2025).

The main goal of this paper is to analyze the performance of the proposed graph construction algorithm in a dynamic environment compared to existing methods such as RRT* and D*Lite in terms of average progress toward the goal, success rate, path efficiency counting nodes.

To achieve this goal, the following objectives are solved:

1. Development of a graph construction algorithm taking cycle detection
2. Modeling a dynamic environment with static and moving obstacles
3. Training agents to identify promising nodes and break out of loops
4. Comparative analysis of the proposed method's effectiveness with the RRT* and D*Lite algorithms across a number of metrics, including goal achievement, path length, and computational cost.

Materials and research methods. The simulation was built using the Pygame library and PyTorch for building a hybrid algorithm. Tests were run over 40 epochs, each with different city structures, starting and ending points, and dynamic obstacles (Paszke et al., 2019). Three algorithms were run each epoch: hybrid path construction, RRT*, and D*Lite.

The simulations are conducted in a procedurally generated two-dimensional urban environment, representing a square region 40 by 40 units. The space is discretized onto a grid with a step of 3 units, at the nodes of which built-up areas and open spaces are formed, this ensures the existence of the path from origin to the endpoint.

Static obstacles are placed with a probability of 0.78 in the corresponding cells and are modeled as rectangles with a random width and length from the range [1.0, 2.4]. Open spaces are additionally introduced with a probability of 0.10 and sizes from the range [2.0, 4.0]. This generation creates diverse topologies with narrow passages and open spaces.

Each scenario is completely determined by a fixed random seed, ensuring reproducibility while maintaining map diversity between epochs. For each scenario, the agent's starting position is randomly selected from the free space, at least 2 units away from the map boundaries. The target position is also selected from the free space, with a minimum distance of at least 10 units (half the environment radius) between the start and target. This ensures that the task requires global route planning rather than local obstacle avoidance. If a suitable target is not found within 1000 attempts, a fallback point selection strategy is used. The duration of one epoch is up to 40 seconds at a frequency of 60 FPS (up to 2400 steps). The agent's observations are based on 4 LiDAR beams for static obstacles (range 6.0), 4 beams for dynamic objects (range 5.0).

All methods (CAFE, RRT*, D* Lite) are evaluated under identical conditions, including identical maps, start-destination positions, and dynamic obstacle configurations. Each experiment includes 40 independent scenarios, each using 32 agents.

For RRT* and D* Lite, each agent independently plans a path based on locally available information. In contrast, CAFE uses a single learned policy applied to all agents without explicit replanning. RRT* was configured with a maximum of 3,000 tree expansion iterations per agent, a step size of 1.8 units, and a rewiring radius of 3.5 units. D*Lite operated on a grid with a cell resolution of 0.8 units and updated affected cells incrementally each simulation frame as moving obstacles altered grid costs.

The RRT* algorithm performs stochastic search tree construction and subsequent path optimization through a rewiring procedure, ensuring asymptotic optimality with limited adaptation to a dynamic environment. D* Lite is an incremental heuristic search algorithm that updates previously computed shortest paths as the environment changes, ensuring efficient replanning, but is sensitive to highly dynamic obstacles.

All methods operate under the same constraints: the maximum episode length is 2400 steps, and goal achievement is determined by reaching a radius of 1.5 units from the target point. Results are averaged across all agents and scenarios.

Table 1. The evaluation metrics

Metric	Description
Average Progress	Average normalized decrease in distance to the goal
Success Rate	Percentage of agents that reach the goal
Steps	Average number of steps required to reach the goal
Collisions	Average number of interactions with dynamic obstacles
Stability (Std)	Standard deviation of progress across agents
Graph Nodes	Average number of nodes generated during navigation

The Table 1. Demonstrates the metrics used after each epoch which will be reviewed in the paper. The Graph Nodes metric reflects the computational complexity of the methods: for CAFE, it is the size of the generated navigation graph; for RRT*, it is the number of search tree nodes; for D* Lite, it is the number of nodes involved in the replanning process.

32 agents for their state uses all known graph properties, but the one novel use is the weight of the node, which corresponds to the openness (Lowe et al., 2017). Before an agent can decide where to go next, it has to discover potential spots it may choose to visit. When constructing the hypothetical nodes of the graph, they are marked as unexplored and serve as targets for agents to visit. The order of visits is determined by a weight, which includes parameters such as proximity to the target and path openness. If there are no objects around a point in the agent's field of view, it will have a higher weight. This weight also depends on proximity to other nodes; the closer the node, the worse the situation, requiring the creation of a wider graph. Upon reaching a new node, the agent draws spokes around itself in eight directions, horizontally, vertically, and diagonally, at a distance of 3.2 conventional units. If an object occupies the node's position, the node does not appear. In other words, a tree is constructed, from which further trees will then be constructed. The vectors of the probable nodes if hits the building is removed. The rest of the nodes are indicated as unvisited, which forces the agents to explore that node. The priority is set due to the weight of the node, which analyze how open it is from the view of the closest agent, the distance towards the goal and the closeness to another node. The further node with no building around has high priority. If agent visited node, it is marked as visited, which plays role as a free passage for the agents which moves behind it.

A neural network for the avoidance processes wall LIDAR 4 rays through LayerNorm layers, then merges them with a 16-dimensional navigation context vector, which includes goal direction, nearest node attributes, status flags. The quarter of the best results proceed further for the training, the reinforcement learning is used.

The central innovation to form closed loops around objects, in order for the agents to not create extra calculations. The ring structure forces other agents to get away from the ring in order to explore more area around the existing loop in order to create a new loop, and so on until it creates a rectangular path to the goal. In this state, a navigation module without knowledge of the cycle`s topology can send the agent in a closed orbit instead of moving toward the goal. The cycle detection function is called each time the agent arrives at a new node and checks for the presence of such a ring for each building in the environment. For a building closed by nodes, the algorithm collects all visited nodes within a radius of 4.5 units. If there are at least three such nodes, they are considered participants in the cycle.

Polar angle for sorting the nodes of a cycle around a building (Harik & Korsaeth, 2019):

$$\theta_k = \text{atan2}(n_{y_k} - c_y, n_{x_k} - c_x) \tag{2}$$

where:
θ_k - polar angle from node k relative to the center of the building;
n_{x_k}, n_{y_k} - coordinates of the k node;
(c_x, c_y) - coordinates of the center of the object.

After sorting by θ_k, the nodes are connected sequentially: the edge between $keys[i]$ and $keys[(i+1) \pmod N]$. All nodes of the cycle are marked with the corresponding flag. For each node, whether it is a frontier node is determined: the "outward" unit vector from the center of the building to the given node is calculated, then a check is made to see if there is at least one of the children spokes whose direction forms an angle of less than 72.5 degrees with the "outward" vector (the dot product is greater than 0.3), provided that the path to this spoke does not intersect any buildings. The frontier-nodes when in close proximity and the agents sees the free passage doesn't complete the loop by itself, the node is connected anyways. Such is required to optimize the time for the agents to create a path with less movement. The Figure 1. shows the node creation where nodes are sorted by the size, the bigger the more weight it gets. Blue points probable future nodes, yellow visited nodes. Blue point

initial point, red point is the target.

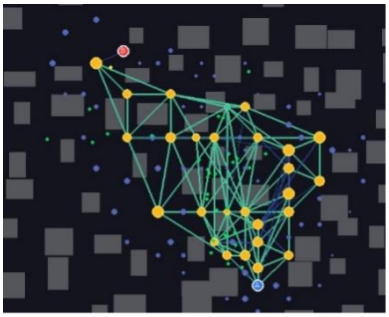

Figure 1. The node and edges creation

The Table 2. Shows the different types of the dynamic objects added to the testing. They are slower than agents and required to create an obstacle for the different approaches to create a path in a dynamic situation. There are 18 moving objects each epoch.

Table 2. Dynamic obstacle types

Type	Count	Speed (u/frame)	Behavior
Patrol	5	0.10	Fixed A to B route
Roamer	6	0.08	Random walk, wall bounce
Chaser	4	0.07	Pursues nearest agent; idles 2–5 s periodically
Blocker	3	0.04	Slow drift with random pauses at junctions

Figure 2. shows the implementation of the dynamic objects, which have different behavior, the most intensive learning creates the type “Chaser”, which always forces nodes to retreat to the previous node.

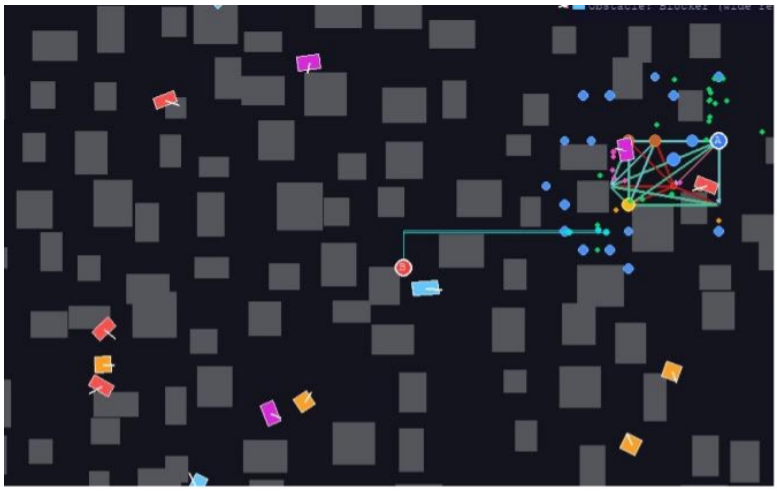

Figure 2. The implementation of dynamic objects

The neural system has two modes in its architecture, which contains several numerical parameters (Nguyen et al., 2020). The first stream operates on data from four distance sensors, normalized distances to the nearest walls in the forward, forward-right, forward-left, and backward directions. It is a single linear layer with four inputs, thirty-two outputs, and a ReLU activation function (Khan et al., 2020).

The second stream works with navigation features: the direction and distance to the target, properties of the nearest graph node (direction, distance, weight), closeness to the buildings, and four agent state flags. Besides a ReLU activation, a single linear layer with twelve inputs and eighty outputs forms its basis. The two streams` outputs of 112 neurons are combined and passed through two hidden layers of 192 neurons each. Subsequently, the Gaussian policy parameters are constructed: a vector of mean values and a learnable parameter, the logarithm of the standard deviation. The splitting of the streams was motivated by experiments: training a single-threaded network with the same number of parameters resulted in a final average progress of 0.52 only, whereas a dual-threaded network gave 0.71 (Haarnoja et al., 2018).

Results. The Table 3. shows the results after 40 epochs training. Basically, RRT* shows a non-zero result over 40 epochs, since a hypothetical path is initially set, and they move towards it and do not get stuck in one place, like other models, which sometimes often close them into a corner when objects behave aggressively. The cost is efficiency averaged 1409 steps per epoch, more than twice paper`s proposed work 682 steps when successful. D*Lite`s low results can be explained that it always tries to reconstruct the path, even the slight connection to the moving object, freezes the agents in order to create a new path, and if the moving object leaves the previous point without returning to that position (Chen et al., 2019).

Table 3. Summary of results across 40 epochs

Algorithm	Mean average progress	Peak average progress	Mean % reached	Mean steps when reached	Mean collisions /agent
Dynamic node creation	0.277	0.792	13.3%	682	8.2
RRT*	0.373	0.536	29.5%	1409	7.3
D*Lite	0.033	0.099	2.3%	2311	0.1

The proposed algorithm with RRT* and D*Lite across 40 epochs with several static and dynamic obstacles. The results shows that RRT* performs better than proposed model, however it works faster with no crowded environmental. The proposed model found more solution from the aggressive moving objects more than in 30% of the occasions. The D*Lite overall showed the poor realization with chasers. The idea was that to create a passage from the hard situations, because the Manhattan path with the neural activation anyway always finds the way, and tried to find the safest path, not to collide with other objects (Chiang et al., 2019).

The results for average goal achievements show that RRT* is superior, but when looking at the results for goal achievements, we can compare graph construction and time. The average number of nodes for RRT* was 1409, while the proposed system required 682 nodes, which is twice as many. Therefore, we can conclude that for studying goal achievements, RRT* constructs significantly more nodes, which require more computation, and shows that it avoids obstacles with more nodes than the proposed method. For dense locations, the hybrid method would be better, as it avoids obstacles better and does not circle around dynamic obstacles less often (Kiran et al., 2021).

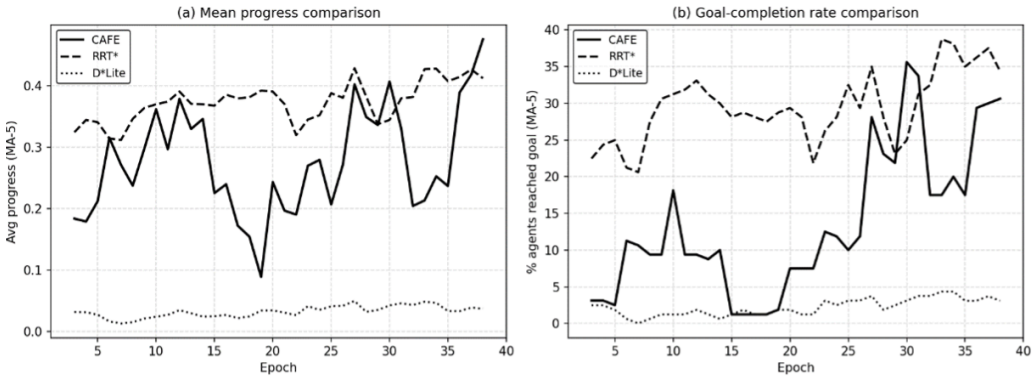

Figure 3. The comparison table of the three models

In the Figure 3. denote “CAFE” as the paper`s proposed model about node creation, which uses Cycle-Aware Frontier Exploration. Less than half of the agents exactly in 30 epochs completed the goal. The average when agent got the goal were in the 47%, excluding zero results. The clear rectangular path if is not locked the agents get to the goal 100%, however the simulations most of the time created a temporary barrier to the goal. The not linear results indicated that some of the occasions might be hard to construct the passage or impossible for the agents.

The average number of cycles detected per epoch was 8.2, with the average number of graph nodes stabilized at around 39 per epoch, with a maximum of 78. During training, the mean value was 0.25, with a standard deviation of 0.28. Such high variability is typical for problems where the complexity of the environment varies from epoch to epoch: some maps show a straight path without obstacles, while others are full of dense buildings and intersecting obstacle trajectories.

To further characterize the structural differences between the three algorithms, a k-means clustering analysis (k = 3) was applied to the combined set of 120 epoch-level observations (40 per algorithm) using two features: average normalized progress and goal success rate. Features were standardized prior to clustering. The resulting cluster assignments and convex hulls are shown in Figure 4.

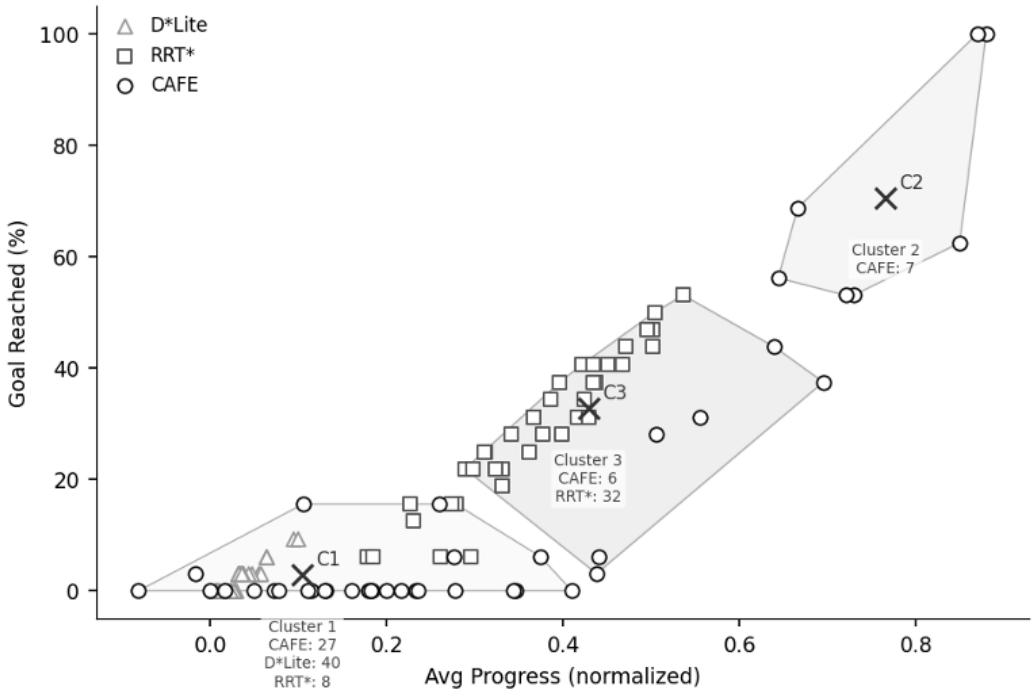

Figure 4. K-means clustering (k=3) of all epoch observations across CAFE (circles), RRT* (squares), and D*Lite (triangles)

The clustering reveals a clear separation between the three algorithms' behavioral profiles. Cluster 1 (centroid: progress = 0.10, success = 2.9%) captures epochs of low-performance operation and contains all 40 D*Lite epochs together with 27 CAFE epochs, predominantly those in which dense obstacle configurations prevented graph construction from converging. The 8 RRT* epochs in this cluster correspond to the algorithm's weakest map seeds. Cluster 3 (centroid: progress = 0.43, success = 32.8%) is dominated by RRT* (32 of 40 epochs), reflecting the algorithm's characteristically moderate but consistent progress across map types, with 6 CAFE epochs that achieved comparable mid-range performance.

Cluster 2 (centroid: progress = 0.77, success = 70.5%) is composed exclusively of 7 CAFE epochs all cases where the cycle-aware graph successfully constructed an unobstructed corridor to the goal. No RRT* or D*Lite epochs appear in this high-performance cluster, which confirms that CAFE is the only algorithm capable of reaching the high-progress, high-success-rate regime in this experimental setting. The absence of RRT* from Cluster 2 is notable: despite its higher mean success rate overall, RRT* never achieves the combination of high progress and high success rate that CAFE attains when its graph construction succeeds. This suggests that CAFE's topology-driven path structure, when effective, produces qualitatively superior navigation performance compared to stochastic tree expansion.

During the training period, the frequency of collisions of agents with moving obstacles remained almost constant in absolute frequency, but the types of collisions changed. Since the neural network had not yet recognized the dynamic range detection signal as a threat, agents collided with obstacles in the early stages mainly because they failed to react. In later stages, the main cause of accidents became the situations around turns - when an attempt to avoid a collision with a moving object ended in a collision with the wall of a building (García & Fernández, 2015). Starting from epoch 50, the total time spent on avoidance decreased from 7% to 4%, this change is due to the fact that the model has learned to avoid obstacles more successfully.

Discussion. Unlike existing mapping methods, the proposed algorithm provides a new method for target detection. While many modern methods require massive amounts of cumulative sensor data, the proposed method builds closed loops. If three or more nodes are not connected by a distance of 4.5 conventional units and there are no static objects within the nodes, the system identifies them as static objects. If nodes are present within the nodes, it is considered a dynamic object. This not only allows the method to be used with minimal sensor equipment, but also makes it applicable to locations that the agent sees for the first time.

The outcomes show the advantages and disadvantages of the suggested strategy. Besides the model, the fact that agents are able to independently bypass objects to restore the initially predicted path means that for production tasks, a path will be built quickly, and if the object is surrounded by other walls, then the agent will have at least some chance of finding a path if one exists. The system creates nodes and saves them like a map, that can be used for the future work. The RRT* creates path from the beginning and tries to get to the point the faster, the frontier-oriented model expands when several agents work, and when the building is located it creates the closed loop which indicates agents to move away from that instance (Cadena et al., 2017). Another usage of the model, that the path sometimes can be closed, and the algorithms tries to investigate the area, when it opens it create the path.

To complicate the situation, dynamic objects were also introduced into the system, and to bypass them without disrupting the entire architecture of the long-term structure, a short-term algorithm for bypassing such an object was created. The graph describes the static structure of buildings, not the immediate state of space, therefore long-term path planning via it is resilient to the presence of moving objects. The model's structure is divided into several parts. Instead of a single module responsible for everything, several modules were identified, each responsible for its own specific operation. Initially, a perpendicular long-term path is constructed, which is optimized each time the agent gets closer to the goal. The neural module works on the current situation when

encountering an object, meaning it must first exit the situation and return to the original path. This structure is easier to construct and correct if any issues in the agent's behavior are identified (Zhou et al., 2025).

By assigning vectors to objects, such as velocity and viewing angle, for both the agent and objects, it creates a huge developmental direction. That is, if the agent sees a moving object, using its motion vector and assuming uniform motion, it can identify the zone it will be in in two or three seconds. Consequently, it can avoid this zone, avoiding overloading the neural system more than hypothetically necessary. The proposed structure also operates by anticipating a future structure, rather than purely reacting to the current situation. Such neural avoidance system is always required, not only because of the objects it might face during the path creation, but also to negate the exploitation of the path, that something can use the logic of the path creation in order to interrupt the agent’s movement.

Conclusion. The study examines the topology-based problems for the real world`s scenarios. Many existing works pay attention to the known situation, and the frontier-oriented model helps to investigate the unknown environmental and to save the data of the path, that can be used for the future agents to move on this path. However, some structures might change over time and anyway the neural activation of the obstacle avoidance helps to get out from the closed loop to get to another loop, or to find a way in order to create new nodes (Hossain et al., 2024).

Every urban structure is unique and constantly changing and to get the one fully working navigation system becomes a hard task, The cycle detection proposed in work creates several nodes and joins them in order to track the objects agents face and to get out from the origin point and get each time close to the target.

Several limitations of this study should be highlighted, which could pose challenges in real-world settings. While simplifying all objects to trivial parallelepipeds seems logical, this overly simplified geometry could pose challenges for heuristic node generation algorithms. Secondly, the study's results demonstrate a high variance (0.28 on average) across all epochs. Each epoch generated different situations, constructing structures with highly variable density, and the distances to moving objects created a unique scenario in which the results were not always correlated. Future experiments should consider a more controlled environment with some opportunity to fixate on a single variable to track how a specific factor influences the overall performance of the agents.

The obtained experimental results demonstrate that the proposed cycle-aware graph construction method achieves improved navigation behavior in dynamic environments compared to baseline approaches. In particular, the method reaches a higher average progress toward the goal (≈ 0.277 mean progress) and a peak success rate of 13.3. When the cycle-aware graph succeeded in constructing an unobstructed corridor, agents reached the goal in an average of 682 steps, less than half the 1,409 steps required by RRT*. D*Lite achieved only 2.3% mean success rate under the same dynamic conditions. The proposed approach demonstrates a consistent advantage in path efficiency: when successful, it requires significantly fewer computational steps than RRT*, indicating more compact and targeted graph structures. These results confirm that the main objective of the study, to evaluate the effectiveness of the proposed graph construction algorithm in dynamic environments, has been achieved under the defined evaluation metrics, namely average progress, success rate, path efficiency (node count), and computational stability.

The work could be improved in several ways. After the simulation, it's worth conducting a real experiment to confirm how it works not with black and white plain dimensions, but with real terrain, where the ground isn't always flat. In addition to developing this algorithm on a real machine, such as a Raspberry Pi, it's also necessary to test the processor load and whether it will work under maximum stress. It's also worth considering creating more path blockers and their types. The algorithm was built on creating loops, which in the 3D model will appear as unvisitable locations and will be marked as lines on a plane, which can be developed into a volumetric object.

References

Cadena, et al., 2017 - Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J.& Leonard, J. J. (2017). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on robotics, 32(6), 1309-1332. https://doi.org/10.1109/TRO.2016.2624754

Chen, et al., 2019 - Chen, C., Liu, Y., Kreiss, S., & Alahi, A. (2019, May). Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In 2019 international conference on robotics and automation (ICRA) (pp. 6015-6022). IEEE. https://doi.org/10.1109/ICRA.2019.8794134

Chiang, et al.,2019 - Chiang, H. T. L., Hsu, J., Fiser, M., Tapia, L., & Faust, A. (2019). RL-RRT: Kinodynamic motion planning via learning reachability estimators from RL policies. IEEE Robotics and Automation Letters, 4(4), 4298-4305. https://doi.org/10.1109/LRA.2019.2931199

Garcıa& Fernández, 2015 - Garcıa, J., & Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research, 16(1), 1437-1480. https://www.jmlr.org/papers/volume16/garcia15a/garcia15a.pdf

Haarnoja, et al., 2018 - Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., ... & Levine, S. (2018). Soft actor-critic algorithms and applications. https://doi.org/10.48550/arXiv.1812.05905

Harik& Korsaeth, 2019 - Harik, E. H. C., & Korsaeth, A. (2019). The Heading Weight Function: A Novel LiDAR-Based Local Planner for Nonholonomic Mobile Robots. Sensors, 19(16), 3606. https://doi.org/10.3390/s19163606

Hossain, et al., 2024 - Hossain, J., Faridee, A. Z., Roy, N., Freeman, J., Gregory, T., & Trout, T. (2024, October). Toponav: Topological navigation for efficient exploration in sparse reward environments. In 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 693-700). IEEE. https://doi.org/10.1109/IROS58592.2024.10802380

Khan, et al., 2020 - Khan, A., Ribeiro, A., Kumar, V., & Francis, A. G. (2020). Graph neural networks for motion planning. https://doi.org/10.48550/arXiv.2006.06248

Kiran, et al., 2021 - Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., & Pérez, P. (2021). Deep reinforcement learning for autonomous driving: A survey. IEEE transactions on intelligent transportation systems, 23(6), 4909-4926. https://doi.org/10.1109/TITS.2021.3054625

Lowe et al., 2017 - Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems, 30. https://doi.org/10.48550/arXiv.1706.02275

Nguyen, 2020 - Nguyen, T. T., Nguyen, N. D., & Nahavandi, S. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. IEEE transactions on cybernetics, 50(9), 3826-3839. https://doi.org/10.1109/TCYB.2020.2977374

Noreen, et al., 2019 - Noreen, I., Khan, A., Asghar, K., & Habib, Z. (2019). A path-planning performance comparison of RRT*-AB with MEA* in a 2-dimensional environment. Symmetry, 11(7), 945. https://doi.org/10.3390/sym11070945Paszke, et al., 2019 - Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G.& Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32. https://doi.org/10.48550/arXiv.1912.01703

Waga et al., 2025 - Waga, A., Benhlima, S., Bekri, A. et al. A survey on autonomous navigation for mobile robots: From traditional techniques to deep learning and large language models. J. King Saud Univ. Comput. Inf. Sci. 37, 198 (2025). https://doi.org/10.1007/s44443-025-00216-x

Zhou,et al., 2025- Zhou, J., Yang, H., Shen, J., & Zhu, L. (2025). Indoor navigation map design based on spatial complexity. Cartography and Geographic Information Science, 52(1), 69-81. https://doi.org/10.1080/15230406.2024.2339296

Zhu et al., 2025 - Zhu, Y., Wan Hasan, W. Z., Harun Ramli, H. R., Norsahperi, N. M. H., Mohd Kassim, M. S., & Yao, Y. (2025). Deep Reinforcement Learning of Mobile Robot Navigation in Dynamic Environment: A Review. Sensors, 25(11), 3394. https://doi.org/10.3390/s25113394