I.J. Intelligent Systems and Applications, **2023, 6, 51-64**
Published Online on December 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2023.06.05

An Improved Sampling Dijkstra Approach for Robot Navigation and Path Planning

Ayman H. Tanira*
Computer Science Department, Palestine Technical College, Deir El-Balah, Palestine
E-mail: atanira@ptcdb.edu.ps
ORCID iD: https://orcid.org/0000-0003-4793-9341
*Corresponding author

Iyad M. I. AbuHadrous
Engineering Department, Palestine Technical College, Deir El-Balah, Palestine
E-mail: ihadrous@ptcdb.edu.ps

Received: 10 July 2023; Revised: 02 September 2023; Accepted: 15 October 2023; Published: 08 December 2023

Abstract: The task of path planning is extremely investigated in mobile robotics to determine a suitable path for the robot from the source point to the target point. The intended path should satisfy purposes such as collision-free, shortest-path, or power-saving. In the case of a mobile robot, many constraints should be considered during the selection of path planning algorithms such as static or dynamic environment and holonomic or non-holonomic robot. There is a pool of path-planning algorithms in the literature. However, Dijkstra is still one of the effective algorithms due to its simplicity and capabilities to compute single-source shortest-path to every position in the workspace. Researchers propose several versions of the Dijkstra algorithm, especially in mobile robotics. In this paper, we propose an improved approach based on the Dijkstra algorithm with a simple sampling method to sample the workspace to avoid an exhaustive search of the Dijkstra algorithm which consumes time and resources. The goal is to identify the same optimal shortest path resulting from the Dijkstra algorithm with minimum time and number of turns i.e., a smoothed path. The simulation results show that the proposed method improves the Dijkstra algorithm with respect to the running time and the number of turns of the mobile robot and outperforms the RRT algorithm concerning the path length.

Index Terms: Dijkstra, Improved Dijkstra, Path Planning, Robot Navigation, Mobile Robot, Shortest Path.

1. Introduction

In the last few decades, mobile robots such as Autonomous Vehicles (AVs) have shown tangible results and have been successfully adapted for different tasks in several areas such as military, security, and industry to accomplish unmanned tasks [1]. During navigational missions, robots utilize their capabilities to model the environment, localize position, control motion, determine obstacles, avoid collisions, and navigate in an environment that varies from simple to complex. Machine learning can effectively increase the intelligence of mobile robots leading to more productivity and reliability during the navigation process. Mobile robots navigate in their complex environments using a path-planning algorithm [2]. Thus, one of the main essential issues that have to be solved before autonomous mobile robots start the navigation process is path planning [3].

The task of path planning is the identification of a safe, efficient, and collision-free route in the workspace of the robot to navigate from the source point to the target point [4]. Path planning is an essential task of the robot to accomplish its mission in the working environment. Path planning, workspace precepting, localizing, and motion controlling are the main tasks of mobile robots [5–7]. AV depends on a path-planning algorithm to move from point to point satisfying specific performance criteria such as time, energy, collisions, or distance. To achieve these goals, the selected path planning algorithm should be reliable, efficient, and satisfy the required criteria [8]. Moreover, the path of the robot should be smooth as possible to maintain the resources of the robot such as energy and other resources which maintain the working time of the mobile robot.

The selection of a path-planning algorithm depends on the system of the mobile robot and environment. No path-planning algorithm is appropriate for all applications. While mobile robots are used in different domains, each domain has special characteristics and a specific operating environment. Therefore, the best-performing technique is selected according to the nature of the task and system. To ensure that the autonomous robot transfers through the environment smoothly and

avoids collisions, the path-planning approach should match the configuration space. Accordingly, several approaches are proposed in the literature according to the applicability issues. These issues involve the robot's kinematics, dynamicity of the environment, availability of the resources, capability of the computational resources, and availability of incoming information through sensors and other resources [8].

In path planning, the consideration of available knowledge represents a crucial factor. Ideally, the optimal scenario entails a static environment that is fully observable. However, this ideal condition is not always attainable in real-world scenarios. The environment can exhibit variations ranging from static to dynamic, and its observability may range from full to partial or even unknown. Whereas the mobile robot moves through a specific trajectory, it receives information from different sensors and updates its knowledge to take suitable action. Thus, the path planning must be adapted according to the available information [9].

According to the degree of observability, i.e., the available knowledge of the environment, path planning algorithms are classified into two main categories: global and local. When the environment is fully observable to the mobile robot, i.e., the mobile robot knows all information about the operating environment before starting the navigation process, then the path planning algorithm is global. On the other hand, when the environment is semi-unknown to the mobile robot, i.e., the mobile robot doesn't know most of the information about the working environment, then the algorithm is local [10].

The original Dijkstra algorithm has many limitations for path planning in mobile robots. Firstly, its inability to efficiently handle dynamic environments. The algorithm assumes a static environment where the costs of traversing different nodes or edges remain constant. However, in real-world scenarios, the environment may change over time, such as the presence of moving obstacles, evolving traffic patterns, or varying terrain conditions. The Dijkstra algorithm does not consider these dynamic changes, resulting in suboptimal or even infeasible paths. Secondly, its computational complexity where the algorithm explores all possible paths from the starting point to all other nodes in the graph until it reaches the goal. This exhaustive search makes it inefficient for large-scale environments with a high number of nodes and edges. In particular, the time complexity of Dijkstra's algorithm grows rapidly as the size of the graph increases. For mobile robots operating in real-time environments, such as autonomous vehicles or drones, the Dijkstra algorithm's high computational cost can be a significant drawback. It may lead to delays in path planning, making it unsuitable for applications that require quick decision-making and responsiveness.

This work addresses the problem of running time and the number of turns of the mobile robot when using the Dijkstra algorithm for path planning to make the Dijkstra algorithm applicable and efficient in real-life applications. Instead of the exhaustive search by the original algorithm, the proposed method samples the working environment to eliminate the exploration of paths that don't more likely lead to the goal, thereby reducing the overall running time and improving efficiency.

The goal of this work is to propose an improved Dijkstra algorithm with the same path as the Dijkstra algorithm but with practical running time and fewer turns. The practical and realistic application of this paper focuses on using Dijkstra's algorithm and a mobile robot to determine the optimal path within a working environment, aiming to reach a specific target. The significance of this work lies in addressing real-world challenges and applying the theoretical concepts to achieve tangible results in practical scenarios.

Despite being introduced in the 20th century, the original Dijkstra algorithm continues to garner significant interest from researchers working in the field of mobile robot navigation and path planning. This work distinguishes itself from prior research by utilizing the original Dijkstra algorithm even after sampling the working environment through a straightforward sampling method. The objective is to eliminate a significant number of paths during the navigation process to reduce the consumed time and enhance efficiency.

The main contributions of this paper are as follows:

- Introducing an improved sampling Dijkstra approach for robot navigation and path planning where the proposed method relies on modeling the working environment instead of an exhaustive search to make the solution applicable in real-life applications.
- Formulating the proposed method as a formal algorithm named Improved Dijkstra that merges a simple sampling approach with the traditional Dijkstra algorithm.
- Conducting a set of experiments to measure the performance of the proposed method. The goal is to find the optimal path as possible as the Dijkstra algorithm but with practical running time and fewer turns.

The rest of the paper is organized as follows. Section 2 presents previous work on path-planning algorithms. Section 3 explains the traditional Dijkstra algorithm. Section 4 introduces the proposed technique and methodology. The experimental results and discussions are reported in Section 5. Finally, Section 6 concludes the paper.

2. Related Works

There is a pool of algorithms that have been proposed for path-planning tasks. These algorithms are stemmed from several disciplines where path planning is not limited to mobile robotics. In general, any problem needs a sequence of actions to be taken from an initial point to reach a final point using a path planning algorithm. So, path planning attracted the attention of many researchers.

Many heuristic approaches for global path planning have been proposed. The Dijkstra algorithm is one of the famous approaches [11]. It relies on the graph theory where the problem is modeled as a directed weighted graph consisting of a set of vertices and a set of edges before applying the single-source shortest path to every vertex in the graph. Each weighted edge represents the cost of moving from the first vertex to the second one. We will explain the details of the algorithm in section 3. The A* algorithm is an extended version of the Dijkstra algorithm [12]. It begins from a certain vertex then it updates the weights of the children. The minimum weight through the node's children is used to update the weight of the current vertex until all vertices are traversed. The D* algorithm is proposed for robot path planning [13]. The algorithm begins by determining the position of the robot to determine the search line of the optimal path. Other versions of the algorithm, the field D* and Theta D* are proposed in [14–16]. Rapidly-Exploring Random Trees (RRT) algorithm expands in all directions of the working space to randomly model the environment through a set of vertices. It selects the nearest nodes and finds the path from source to target. It is an efficient algorithm but doesn't guarantee the shortest path [17]. Probabilistic Road Maps (PRM) algorithm intersects with RRT in many characteristics, but it requires a lot of connections between a huge set of states to find a path. RRT and PRM were designed for non-holonomic constraints [18].

On the other hand, there are several approaches stemming from artificial intelligence and machine learning such as Artificial Neural Network (ANN) [19, 20]. A set of Genetic algorithms (GA) is introduced for resolving a path-planning problem [21–23]. Also, Fuzzy Type 1 and Fuzzy Type 2 planning algorithms are found in [24, 25]. ANN approaches try to perceive the environment to map the working space to the behavior space, i.e., a set of actions. Moreover, when merging ANN with Reinforcement Learning (RL) approaches such as Q-Learning (DQN), the agent (mobile robot) can learn more about its environment and can take the appropriate action of movement [20, 26]. In GA approaches, the solutions to the problem are converted to chromosomes from the initial population then a fitness value is generated to determine the basic operations of GA: crossover, mutation, and selection. Finally, GA extracts the trajectory.

Come back to Dijkstra algorithm, the traditional Dijkstra algorithm was proposed by E.W. Dijkstra in 1959 [11]. It is a traditional, well-known, simple, shortest-path algorithm. It was successfully approved in many domains such as mobile robotics, networking, transportation, etc. In this section, we will investigate recent papers based on the Dijkstra algorithm for solving the problem of path planning.

The researchers in [27] introduce an improved Dijkstra algorithm to eliminate the fuel consumption of the vehicle. They reduce the running time of the algorithm by reducing the search area by utilizing a rectangular shape which is adapted based on the dynamicity of the traffic network. The experimental results show increasing efficiency in the proposed model.

The authors in [28] utilize the Dijkstra technique to analyze tree diagrams. Firstly, they construct the tree's nodes using mining blocks. Then, they compute the minimum cost path to transfer the blocks from the current node to the target node. To achieve the shortest path, Euclidian metric and transfer time are adopted to compute the minimum path. They adjust the parameters according to some equations to correct the cost of each block. The results reveal that there is an improvement in the efficiency of mining planning.

The proposed method in [29] aims to trade-off between the limited distance that an electric vehicle can move and the charging demand of the vehicle. The researchers use a dynamic version of the Dijkstra algorithm to identify the shortest path between any two adjacent vertices on the route with variable time on charging and traveling the electric vehicle.

One of the disadvantages of the original Dijkstra algorithm is that it stores data along with the traversed vertices. Thus, to solve this problem, the researchers in [30] provide a hierarchical data storage that consists of a multilayer dictionary. A list data structure is used as a container of all vertices of the graph associated with the first layer that stores for every vertex in the graph, the neighboring vertices. The second layer maintains the distance of the pathway of all neighboring vertices. The proposed multilayer data structure helps the robot to navigate indoors where Global Navigation Satellite System (GNSS) information is not available or not reliable. The experimental results show that the proposed data structure makes the Dijkstra algorithm more efficient in terms of running time and the number of turns.

Other researchers propose a hybrid Dijkstra algorithm with dynamic programming to assist the designers in allocating the appropriate position of towers to expand the transmission lines [31]. In this case, the overall network is modeled as a graph where the transmission stations are the vertices, and the transmission lines are the edges. The goal is to determine the position of new transmission stations with minimum cost. The results explain that the proposed method gives a lower design cost than the traditional transmission lines method.

The Floyd method is also a popular method to determine all-pairs shortest-path with positive or negative weighted graphs without details of the paths themselves [32] whereas the Dijkstra method does not apply to a negative-weighted graph. It outperforms the Floyd method in the single-source shortest path to every vertex in the graph. However, the Dijkstra algorithm suffers from the problem of memory heavy because it computes and maintains information about all possible paths to identify the shortest path from the source to other vertices. Moreover, it suffers from a long running time where the time complexity is $O(n^2)$ where n is the number of vertices. The authors in [8] generalize that the Dijkstra algorithm suites the environment if fully observable i.e., local path planning, and is static. That is, all components of the environment are predefined such as positions of obstacles. However, an improved version of the algorithm can be adapted for dynamic environments and partially to unknown environments. Accordingly, in this work, we try to improve the Dijkstra algorithm to a fully observable environment to determine the shortest path and collision-free by sampling the environment to reduce the workspace and increase the efficiency of the mobile robot during navigation.

3. The Dijkstra Algorithm

The original Dijkstra algorithm is simple and applicable to a directed positive-weighted graph. Thus, to apply the Dijkstra algorithm to determine the shortest-path we need to model the operating environment into a graph $G(V, E)$ where V is a set of vertices and E is a set of positive-connected weighted edges. It finds out the shortest path, d_t between a source vertex, v_s, and all other vertices, v_t, in the graph. Accordingly, the input of the algorithm consists of, V, E, and v_s. The output of the algorithm is the shortest path, d_t, from the source vertex, v_s, to all other vertices, v_t. The computed shortest-path is the one over all possible paths from source, v_s, to targets, v_t and it is the sum of the weights of each pair of vertices in the shortest-path. That is, $d_t = \sum_{i,j} w_{ij}$ where w_{ij} is the weight of the pair (v_i, v_j). Algorithm 1 explains the computation steps of the Dijkstra algorithm.

The Dijkstra algorithm maintains two sets of vertices, V_e and V_u. V_e contains all vertices that their shortest paths from v_s have been determined. V_e is initialized with v_s in the initialization step and it grows through the incoming steps to contain all vertices, V. V_u is the set of vertices that need to compute their shortest path. V_u is initialized with all vertices without the source vertex, i.e., $V - v_s$ and it degraded during the next steps by moving vertices from V_u to V_e in ascending order. That is, the algorithm extracts the vertex with the minimum accumulated weight. V_u becomes an empty set at the termination of the algorithm, i.e., $V_u = \emptyset$. The algorithm performs the relaxation step to determine if the next connected vertex contributes to the shortest path. At the end of the algorithm, the shortest path from the source vertex, v_s, to every other vertex in the graph is determined. The shortest path contains the path sequence of vertices and the computed weight. The main disadvantage of the Dijkstra algorithm is the running time where its time complexity is quadratic with respect to the number of vertices, i.e., $O(n^2)$. In the case of mobile robots where the environment is continuous, it takes a long time to move from the start position to the target position.

4. The Proposed Method

The proposed system consists of five phases, Environment perception, feature extraction, environment modeling, searching shortest-path, and mobile robot navigation as shown in the block diagram in Fig. 1. Each phase performs a specific task and outputs to the following phase as follows:

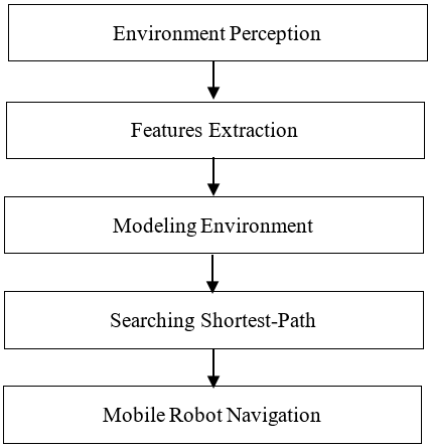

Fig.1. Block diagram of the proposed system method

4.1. Environment Perception

The proposed method is applicable in an environment that is static and fully observable. This means that all aspects of the environment are known and remain constant throughout the planning process. To achieve this, the full information of the environment is obtained before applying off-line shortest-path planning. In order to gather this information, a single vision sensor in the form of an overhead camera can be utilized. This camera is strategically positioned to monitor the entire environment, capturing a comprehensive view of the surroundings. By utilizing this overhead camera, the proposed method ensures that all relevant details and features necessary for planning are captured. The captured information from the overhead camera serves as the input for the off-line shortest-path planning algorithm which processes the gathered data to determine the optimal path or route within the environment. Since the environment is assumed to be static and fully observable, there is no need for real-time updates during the planning phase.

4.2. Feature Extraction

The robot position is estimated at each site by leveraging collected image frames and employing feature extraction and template-matching techniques from the field of image processing. These techniques analyze the visual data captured by the overhead camera, identifying key features and patterns that can be used to determine the robot's position within the

environment as illustrated in Fig 2. Feature extraction involves extracting distinctive characteristics from the image frames that are unique to specific objects in the environment. These features could include edges, corners, textures, or other visual cues. By detecting and extracting these features, the system can identify and localize various elements within the environment. Template matching is a technique used to estimate the robot's position. In this approach, a template or reference image of the robot is created beforehand. During the estimation process, the system compares the template with the captured image frames, searching for regions that closely resemble the template.

By applying these image processing techniques, the system can estimate the robot's position at each site within the environment. Additionally, the system can also identify and track the coordinates of obstacles present in the environment. This comprehensive knowledge of the environment components, including the robot's position and the coordinates of obstacles, allows for effective planning and navigation along the shortest path.

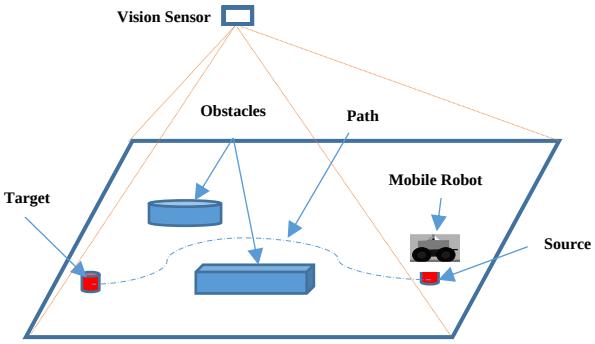

Fig.2. Overall structure of the environment

4.3. Environment Modeling

The proposed system converts the working environment into a grid containing a set of cells. Each cell is surrounded by eight points. From each point, the mobile robot can take one of eight movements: right, left, up, down, right-up, right-down, left-up, and left-down as shown in Fig. 3. Therefore, the proposed system generates eight points from each point as shown in Fig. 3. The reason is that the Dijkstra algorithm needs to know the predecessor of each vertex. Therefore, the point corresponds to a vertex, the movement represents an edge, and the distance between the two points is the weight between the two vertices in the modeling system. Thus, the system converts the environment into a graph $G(V, E)$. Algorithm 2, Improved Dijkstra, illustrates all steps of the proposed algorithm. To generalize the algorithm, we set n as the number of actions and l as the distance to the neighbor vertex. Additionally, step 1, sampling, and step 2, calling the Dijkstra procedure can be implemented to operate concurrently to eliminate the execution time. That is, while the algorithm is sampling the environment, it computes the shortest path.

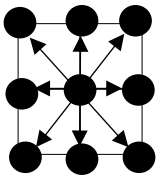

Fig.3. All possible movements

4.4. Searching Shortest-path

The Dijkstra algorithm can be effectively applied to the graph $G(V, E)$ to determine the shortest path from the start position to the target position. In this context, the graph G represents the sampled working environment obtained from the previous step. It consists of a set of points that accurately represent the workspace, with edges (E) connecting these points and defining the relationships between them. To apply the Dijkstra algorithm, each point in the graph $G(V)$ represents a node, and the edges (E) between the points represent the connections or paths between these nodes. Additionally, a set of weights is associated with each edge, indicating the cost or distance required to traverse from one point to another. During the exploration process, the algorithm selects the node with the lowest cost as the current node and expands its neighboring nodes, recalculating their costs if a shorter path is found. This process continues until the target node is reached or all reachable nodes have been explored.

By utilizing the Dijkstra algorithm on the graph $G(V, E)$, the proposed method can determine the shortest path from the start position to the target position within the sampled working environment. This allows the system to efficiently plan the optimal route for the robot, taking into account the costs associated with traversing between different points.

Algorithm 1: Dijkstra algorithm

Input:
A graph $G(V, E)$.
The source vertex v_s.
The target vertex v_t.

Output:
The shortest-path from v_s to v_t (sequence of vertices and the computed weight, $d_t,\ \pi_t$).

1. Initialization:
$V_e = v_s$
$V_u = V - v_s$
for every vertex u in V
    $d_u = \infty$
    $\pi_u = NULL$
$d_s = 0$

2. Processing:
$u = extract\_min(w_{vu})$
$V_e = V_e \cup u$
while $V_u \neq \emptyset$
    for every vertex $v \in adj[u]$
        if $d_v > d_u + w_{vu}$
            $d_v = d_u + w_{vu}$
            $\pi_v = u$

3. Termination:
return $d_t,\ \pi_t$

Algorithm 2: Improved Dijkstra algorithm

Input:
An environment E
The source vertex v_s.
The target vertex v_t.
Length of cell l

Output:
The shortest-path from v_s to v_t (sequence of vertices and the computed weight, $d_t,\ \pi_t$).

1. Sampling:
$V = V \cup v_s$
$V_s = V_s \cup v_s$
for every vertex v_u in V_s:
    generate n vertices, v_i from v_s w. r. t. l
    for every vertex v_i:
        if v_i coordinates are unacceptable
            ignore v_i and continue.
        else
            compute weight w_{ui}
            $adj[v_u] = adj[v_u] \cup v_i$
            $V = V \cup v_i$
            $V_s = V_s \cup v_i$
    $V_s = V_s - v_u$

2. Processing:
$d_t,\ \pi_t$ = Dijkstra $(G(V, E), v_s, v_t)$

3. Termination:
return $d_t,\ \pi_t$

4.5. Mobile Robot Navigation

Once the optimal shortest path has been extracted by the Improved Dijkstra algorithm, the mobile robot is capable of smoothly moving from the start position to the target position. This movement takes place within a continuous working environment, allowing the robot to navigate seamlessly through its surroundings. The extracted shortest path obtained from the Improved Dijkstra algorithm offers optimality in terms of both the path length and the number of turning angles. This means that the path chosen for the robot ensures the shortest distance between the start and target positions, while also minimizes the number of the required turns along the way. By doing so, the robot can follow a more direct and efficient trajectory, saving time and energy during its movement.

One notable advantage of the Improved Dijkstra algorithm is its improved running time compared to the traditional Dijkstra algorithm. This improvement is achieved through various optimizations and enhancements made to the algorithm's implementation. The algorithm can calculate the shortest path more quickly allowing for real-time or near real-time planning and navigation of the mobile robot. Overall, the combination of the Improved Dijkstra algorithm and the mobile robot's ability to follow the optimal shortest path guarantees efficient and precise movement and reduces both the path length and the number of turning angles.

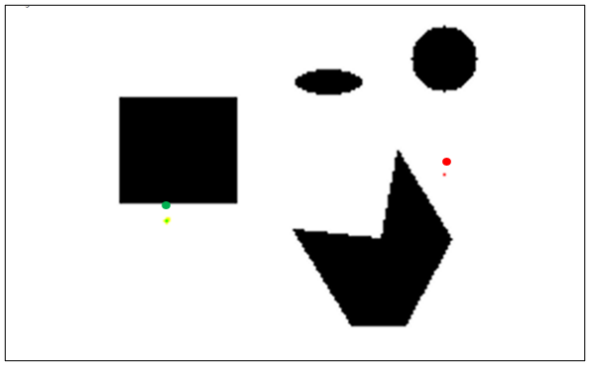

Fig.4. Components of the environment

5. Experimental Results and Discussions

In this section, we present the simulation experiments on path planning analysis using the proposed method. We compare the proposed method with the traditional Dijkstra algorithm [11] and the RRT algorithm [17]. The RRT algorithm is rapid in finding the path from the source to the target but doesn't guarantee the shortest path. The proposed algorithm was implemented in Python 3.10 on a system with an Intel CORE i5, 8GB RAM, and Windows 10 operating system.

5.1. Environment

The simulated environment consists of a rectangular space with 2D, x-axis, and y-axis (250*150). A set of random obstacles are placed randomly in the environment to test the performance of the extracted path and show the movement of the mobile robot as illustrated in Fig. 4. Our environment is fully observable, i.e., we know all information about the environment including the coordinates of the obstacles, the start position, and the final position of the mobile robot. In each experiment, we change the start position and the target position to make the proposed system extracts different shortest paths.

5.2. Evaluation Metric

The efficiency of the proposed system is evaluated through the utilization of three key metrics: the running time, the number of turns, and the path length. These metrics provide valuable insights into the performance and effectiveness of the system in achieving optimal navigation.

- The running time metric measures the total amount of time required for the mobile robot to travel from the start position to the target position along the extracted path. It considers factors such as computational time for planning, decision-making, and actual physical movement of the robot. By assessing the running time, we can gauge the efficiency of the system in terms of its speed and responsiveness during navigation.
- The number of turns metric quantifies the total count of direction changes made by the mobile robot as it traverses along the path. Each turn represents a change in the robot's heading or orientation. Minimizing the number of turns is desirable as it reduces unnecessary rotations, resulting in smoother and more efficient movement. By evaluating this metric, we can assess how effectively the proposed system optimizes the trajectory to minimize turning angles, thus conserving resources and enhancing overall navigation performance.
- The path length metric represents the total number of pixels encompassed by the extracted path generated by the algorithm. This metric provides insights into the spatial coverage of the path and indirectly reflects the efficiency

of the system in choosing the most direct and shortest route. A shorter path length indicates more efficient utilization of the workspace, resulting in reduced travel distances and potential resource savings for the mobile robot.

5.3. Results and Discussions

Table 1 shows the start position and the target position of the mobile robot over five experiments and the results of the running time, the number of turns, and the path length of all algorithms. These results are depicted in Fig. 5, Fig. 6, and Fig. 7. Moreover, we provide all paths extracted from the three algorithms on the simulated environment. In each experiment, the start position of the mobile robot and the target position are set as in Table 1. Table 1 and Fig. 5 show the running time achieved by the RRT algorithm is the best followed by the Improved Dijkstra. It is clear from the figure that the proposed method outperforms the traditional Dijkstra in all experiments. The reason is that the Dijkstra algorithm exhaustively scans the working environment. Whereas, the proposed method divides the working environment into a set of cells representing each cell as a set of points. That is, it reduces the search area significantly. Fig. 6 indicates that the proposed method outperforms the Dijkstra algorithm in two experiments of five while measuring the number of turns that the mobile robot makes during navigation. The RRT algorithm is the worst one where it has the maximum number of turns. It is an important indication that the turn angle of the mobile robot during navigation consumes the robot's resources such as energy. The proposed method outperforms the traditional Dijkstra algorithm due to the environment sampling and reducing the workspace of the mobile robot. Fig. 7 shows that the Dijkstra algorithm and the proposed method determine the shortest path while the RRT is far from the shortest path.

In a nutshell, the proposed method, Improved Dijkstra, addresses one of the main challenges of the traditional Dijkstra algorithm, which is its long-running time when searching for the shortest path. By implementing various optimizations and enhancements, the Improved Dijkstra algorithm significantly reduces the computational burden and improves the efficiency of the search process. As a result, the shortest path is determined with a comparable level of accuracy to that of the Dijkstra algorithm. In addition to optimizing the running time, the proposed method introduces a sampling technique that plays a crucial role in extracting a trajectory with the minimum number of turns. This aspect of the method has significant advantages for the mobile robot during navigation. By minimizing the number of turns required along the trajectory, the resource utilization of the robot, such as energy consumption and wear on mechanical components, is greatly optimized. This translates into extended battery life, reduced wear and tear, and improved overall efficiency of the mobile robot.

By combining the benefits of the Improved Dijkstra algorithm and the sampling method, the proposed approach enables the mobile robot to navigate in an optimal and resource-efficient manner. It not only ensures the shortest path between the start and target positions but also minimizes unnecessary turns, leading to more efficient resource allocation and enhanced longevity of the robot's components. Finally, the proposed method empowers the mobile robot with a faster and more accurate planning algorithm while taking into account the practical considerations of resource management. This allows for more effective and sustainable navigation, making the method a valuable contribution to the field of mobile robotics.

Table 1. Results of five experiments on running time (Ms), number of turns, and path length (Px)

Experiment	Running Time (ms)			Number of Turns			Path Length (px)		
	RRT	Dijkstra	Improved Dijkstra	RRT	Dijkstra	Improved Dijkstra	RRT	Dijkstra	Improved Dijkstra
1	**6**	309	34	162	**3**	**3**	172	**122**	**122**
2	**4**	341	39	185	**3**	**3**	191	**132**	**132**
3	**6**	385	26	196	21	**12**	208	**142**	**142**
4	**9**	394	43	280	9	**7**	283	**187**	**187**
5	**10**	446	43	236	**5**	**5**	269	**197**	**197**

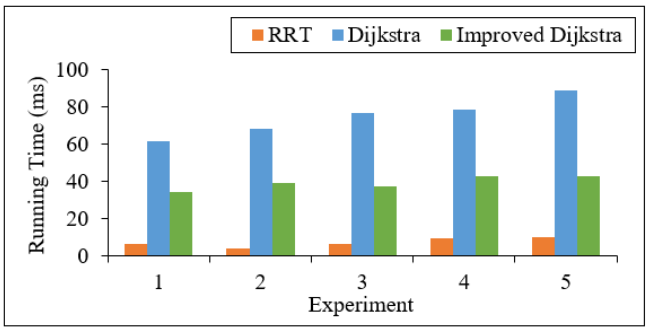

Fig.5. Results on running time (ms), dijkstra is divded by 5

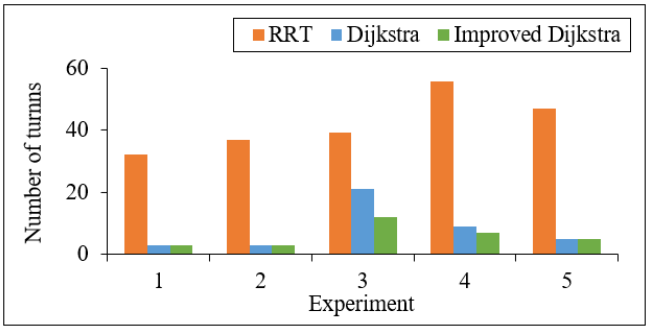

Fig.6. Results on number of turns, RRT is divded by 5

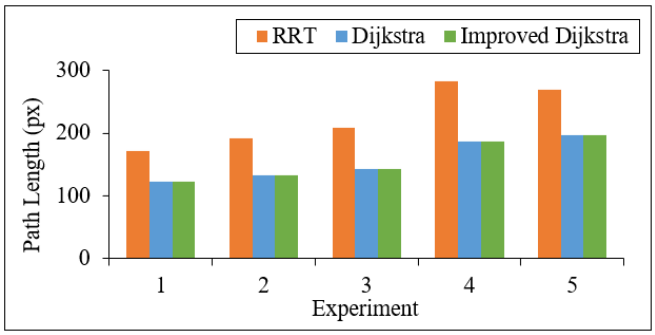

Fig.7. Results on path length in pixels (px)

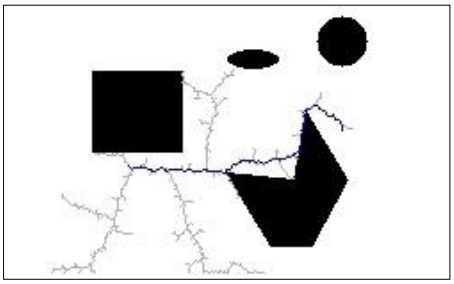

Fig.8. Experemint 1, RRT

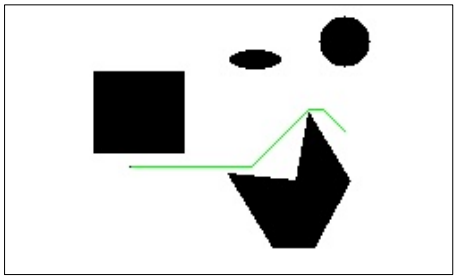

Fig.9. Experemint 1, dijkstra

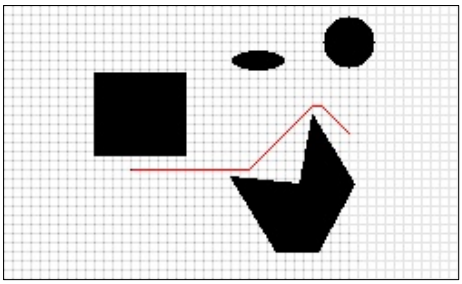

Fig.10. Experemint 1, improved dijkstra

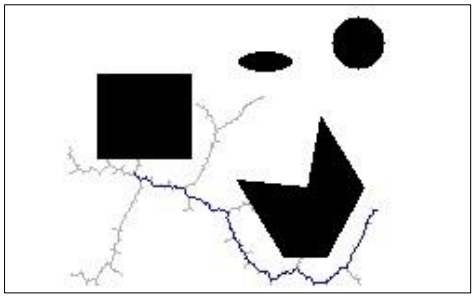

Fig.11. Experemint 2, RRT

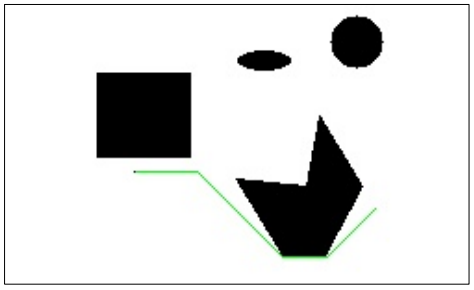

Fig.12. Experemint 2, dijkstra

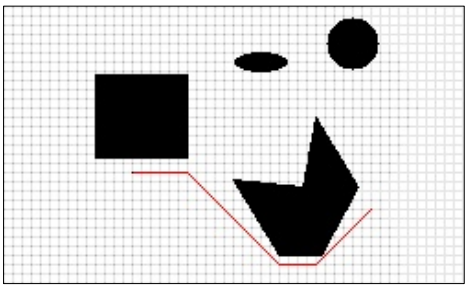

Fig.13. Experemint 2, improved dijkstra

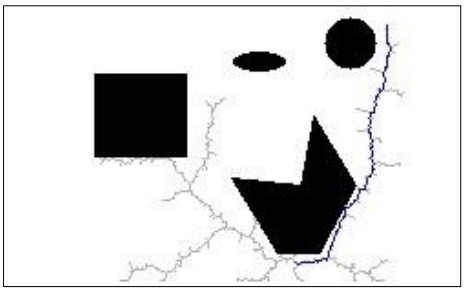

Fig.14. Experemint 3, RRT

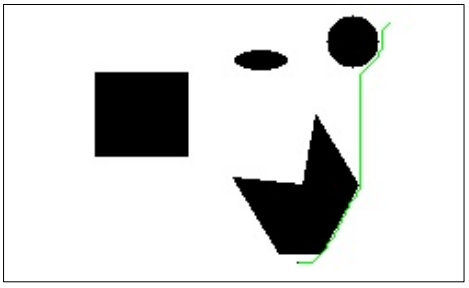

Fig.15. Experemint 3, dijkstra

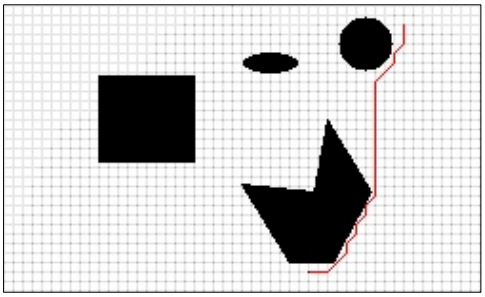

Fig.16. Experemint 3, improved dijkstra

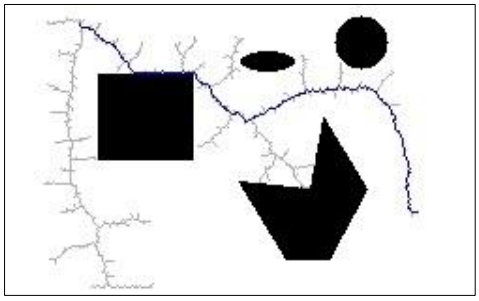

Fig.17. Experemint 4, RRT

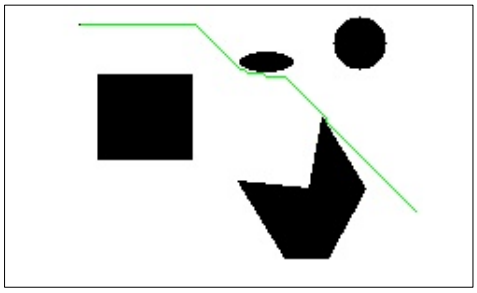

Fig.18: Experemint 4, dijkstra

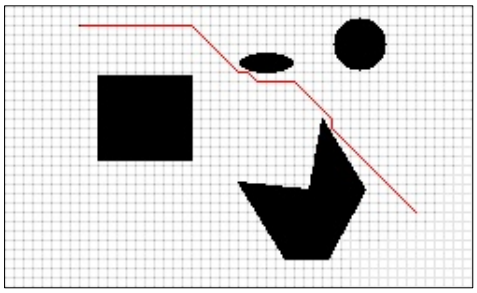

Fig.19. Experemint 4, improved dijkstra

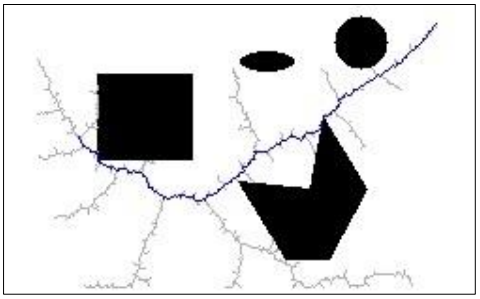

Fig.20. Experemint 5, RRT

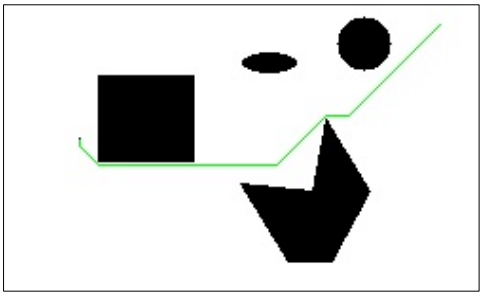

Fig.21. Experemint 5, dijkstra

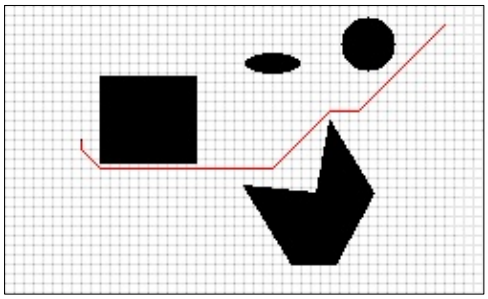

Fig.22. Experemint 5, improved dijkstra

6. Conclusions

In this paper, we propose a novel approach that combines a simple sampling method with the well-established Dijkstra algorithm, aiming to address the primary limitation of the Dijkstra algorithm: its extensive running time for determining the shortest path from the initial position to the final position of a mobile robot. Particularly in continuous environments, where the algorithm exhaustively scans the entire workspace to locate the target position. The objective of the proposed method is to significantly reduce the computation time while maintaining the same level of accuracy in determining the shortest path as the Dijkstra algorithm. To achieve this, we introduce a preprocessing step in which the working environment is transformed into a set of cells, each defined by eight surrounding points connected by links. Consequently, the working environment is represented as a graph comprising vertices (points) and edges (links). This graph is then fed into the Dijkstra algorithm to calculate the shortest path. To validate the effectiveness of our approach, we conducted experimental simulations. The results demonstrate that the widely used RRT algorithm exhibits the fastest running time, but it suffers from suboptimal path length and excessive turns. On the other hand, our proposed method, which we term "Improved Dijkstra", surpasses the traditional Dijkstra algorithm in terms of running time and the number of turns made by the mobile robot, while still maintaining an equivalent path length. By merging a simple sampling technique with the Dijkstra algorithm and presenting empirical evidence of its superior performance compared to both the traditional Dijkstra algorithm and the RRT algorithm, our research provides a scientifically sound and practical solution for improving the efficiency of determining the shortest path in continuous environments for mobile robots.

References

[1] B. Tang, Z. Zhu, and J. Luo, “Hybridizing Particle Swarm Optimization and Differential Evolution for the Mobile Robot Global Path Planning,” Int J Adv Robot Syst, vol. 13, no. 3, 2016, doi: 10.5772/63812.

[2] M. Dirik and A. Fatih Kocamaz, “RRT-Dijkstra: An Improved Path Planning Algorithm for Mobile Robots,” Journal of Soft Computing and Artificial Intelligence, 2020. [Online]. Available: www.dergipark.org.tr/en/pub/jscai

[3] Y. Zhuang, Y. Sun, and W. Wang, “Mobile robot hybrid path planning in an obstacle-cluttered environment based on steering control and improved distance propagating,” International Journal of Innovative Computing, Information and Control, vol. 8, no. 6, 2012.

[4] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, “Heuristic approaches in robot path planning: A survey,” Rob Auton Syst, vol. 86, 2016, doi: 10.1016/j.robot.2016.08.001.

[5] C. Zheng and R. Green, “Vision-based autonomous navigation in indoor environments,” in International Conference Image and Vision Computing New Zealand, 2010. doi: 10.1109/IVCNZ.2010.6148876.

[6] N. Sariff and N. Buniyamin, “An overview of autonomous mobile robot path planning algorithms,” in SCOReD 2006 - Proceedings of 2006 4th Student Conference on Research and Development “Towards Enhancing Research Excellence in the Region,” 2006. doi: 10.1109/SCORED.2006.4339335.

[7] N. B. Sariff and N. Buniyamin, “Ant Colony System for Robot Path Planning in Global Static Environment Faculty of Electrical Engineering,” Selected Topics in System Science and Simulation in Engineering, 2010.

[8] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, “A Survey of Path Planning Algorithms for Mobile Robots,” Vehicles, vol. 3, no. 3, 2021, doi: 10.3390/vehicles3030027.

[9] G. Klančar, A. Zdešar, S. Blažič, and I. Škrjanc, Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems. 2017.
[10] P. Li, X. Huang, and M. Wang, "A novel hybrid method for mobile robot path planning in unknown dynamic environment based on hybrid DSm model grid map," in Journal of Experimental and Theoretical Artificial Intelligence, 2011. doi: 10.1080/0952813X.2010.506283.
[11] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numer Math (Heidelb), vol. 1, no. 1, 1959, doi: 10.1007/BF01386390.
[12] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, 1968, doi: 10.1109/TSSC.1968.300136.
[13] A. Stentz, "Optimal and efficient path planning for partially-known environments," in Proceedings - IEEE International Conference on Robotics and Automation, 1994. doi: 10.1007/978-1-4615-6325-9_11.
[14] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The Field D* algorithm," J Field Robot, vol. 23, no. 2, pp. 79–101, Feb. 2006, doi: 10.1002/rob.20109.
[15] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," Journal of Artificial Intelligence Research, vol. 39, 2010, doi: 10.1613/jair.2994.
[16] K. Xiao, C. Gao, X. Hu, and H. Pan, "Improved Theta*: Improved any-angle path planning on girds," Journal of Computational Information Systems, vol. 10, no. 20, 2014, doi: 10.12733/jcis12043.
[17] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," In, vol. 129, 1998.
[18] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," International Journal of Robotics Research, vol. 20, no. 5, 2001, doi: 10.1177/02783640122067453.
[19] P. Martin and A. P. del Pobil, "Application of artificial neural networks to the robot path planning problem," in Applications of Artificial Intelligence in Engineering, 1994.
[20] C. Li, J. Zhang, and Y. Li, "Application of artificial neural network based on Q-learning for mobile robot path planning," in Proceedings of IEEE ICIA 2006 - 2006 IEEE International Conference on Information Acquisition, 2006. doi: 10.1109/ICIA.2006.305870.
[21] M. Zhao, N. Ansari, and E. S. H. Hou, "Mobile manipulator path planning by a genetic algorithm," J Robot Syst, vol. 11, no. 3, 1994, doi: 10.1002/rob.4620110302.
[22] F. Liu, S. Liang, and D. X. Xian, "Optimal Path Planning for Mobile Robot Using Tailored Genetic Algorithm," TELKOMNIKA Indonesian Journal of Electrical Engineering, vol. 12, no. 1, 2014, doi: 10.11591/telkomnika.v12i1.3127.
[23] H. Shorakaei, M. Vahdani, B. Imani, and A. Gholami, "Optimal cooperative path planning of unmanned aerial vehicles by a parallel genetic algorithm," Robotica, vol. 34, no. 4, 2016, doi: 10.1017/S0263574714001878.
[24] J. Y. Jhang, C. J. Lin, C. T. Lin, and K. Y. Young, "Navigation Control of Mobile Robots Using an Interval Type-2 Fuzzy Controller Based on Dynamic-group Particle Swarm Optimization," Int J Control Autom Syst, vol. 16, no. 5, 2018, doi: 10.1007/s12555-017-0156-5.
[25] T. W. Liao, "A procedure for the generation of interval type-2 membership functions from data," Applied Soft Computing Journal, vol. 52, 2017, doi: 10.1016/j.asoc.2016.09.034.
[26] W. Liu, H. Niu, M. N. Mahyuddin, G. Herrmann, and J. Carrasco, "A Model-free Deep Reinforcement Learning Approach for Robotic Manipulators Path Planning," in International Conference on Control, Automation and Systems, 2021. doi: 10.23919/ICCAS52745.2021.9649802.
[27] D. Guo et al., "A vehicle path planning method based on a dynamic traffic network that considers fuel consumption and emissions," Science of the Total Environment, vol. 663, 2019, doi: 10.1016/j.scitotenv.2019.01.222.
[28] F. R. Souza, T. R. Câmara, V. F. N. Torres, B. Nader, and R. Galery, "Mine fleet cost evaluation - dijkstra's optimized path," Revista Escola de Minas, vol. 72, no. 2, 2019, doi: 10.1590/0370-44672018720124.
[29] S. Shao, W. Guan, B. Ran, Z. He, and J. Bi, "Electric Vehicle Routing Problem with Charging Time and Variable Travel Time," Math Probl Eng, vol. 2017, 2017, doi: 10.1155/2017/5098183.
[30] S. A. Fadzli, S. I. Abdulkadir, M. Makhtar, and A. A. Jamal, "Robotic indoor path planning using dijkstra's algorithm with multi-layer dictionaries," in 2015 IEEE 2nd International Conference on InformationScience and Security, ICISS 2015, 2016. doi: 10.1109/ICISSEC.2015.7371031.
[31] A. H. M. Santos et al., "Optimizing routing and tower spotting of electricity transmission lines: An integration of geographical data and engineering aspects into decision-making," Electric Power Systems Research, vol. 176, 2019, doi: 10.1016/j.epsr.2019.105953.
[32] H. Il Kang, B. Lee, and K. Kim, "Path planning algorithm using the particle swarm optimization and the improved dijkstra algorithm," in Proceedings - 2008 Pacific-Asia Workshop on Computational Intelligence and Industrial Application, PACIIA 2008, 2008. doi: 10.1109/PACIIA.2008.376.

Authors' Profiles

Ayman H. Tanira is a lecturer of computer science at Palestine Technical College-Deir El-Ballah (PTCDB), Palestine. He was granted his bachelor's degree in computer science from Mu'tah University, Jordan and he was the top student. Mr. Tanira obtained his master's degree from Cairo University, Egypt, and also, he was the top student among his colleagues. Mr. Tanira is currently a Ph.D. student of computer engineering at the Islamic University of Gaza (IUGaza). His research focuses on deep learning, information security, and mobile robots. Mr. Tanira published a set of papers in international conferences and journals.

Iyad M. Abuhadrous is an associative professor at the Engineering Professions Department, Palestine Technical College College-Deir El-Ballah (PTCDB), Palestine. Dr. Abuhadrous received his B.S. degree in Electronics Engineering from Yarmouk University, Irbid, Jordan in 1997, and M.S. degree in Robotics from UPMC (Paris VI) and INSTN, Paris-France in the year 2000 and the Ph.D. degree in Robotics and Control from Ecole des Mines de Paris, in 2005. He held several positions such as vice dean for Planning and Development at PTCDB, head of Engineering Professions Department at PTCDB. He has over 20 years of teaching and research experience and has published many papers in national and international conferences and journals. His research interest includes mobile robots, robotics manipulators, automatic control, and Microcontrollers.

How to cite this paper: Ayman H. Tanira, Iyad M. I. AbuHadrous, "An Improved Sampling Dijkstra Approach for Robot Navigation and Path Planning", International Journal of Intelligent Systems and Applications(IJISA), Vol.15, No.6, pp.51-64, 2023. DOI:10.5815/ijisa.2023.06.05