Easy Shopper: Path Planning for Shopping Robot

Pranali Desai
M. Engg. Robotics
University of Maryland
College Park, USA
pranali.yashodhar@gmail.com

Sanket Goyal
M. Engg. Robotics
University of Maryland
College Park, USA
sanket193@gmail.com

Raghav Nandwani
M. Engg. Robotics
University of Maryland
College Park, USA
raghav15@terpmail.umd.edu

***Abstract*—The main element of an automated vehicle is to plan the pathway to reach from a start location to a goal node. There are diverse strategies and applications for this problem. This paper presents a novel method to calculate the path. It is a combination of A* and dynamic A* which is implemented on the common task of the shopping cart. A* is being used for the non-dynamic system which is thought so for the initial stages. Later on, a dynamic system is taken into consideration and thus dynamic A*, i.e. D* has been implemented. Shopping Cart is chosen as the application due to the fact of the increase in time consumed to find the products. Items in the shopping complex are well managed in the data and with the help of that path-planning is done to reach the items to be picked up.**

***Index Terms*—Path-Planning, A*, D*, Smart Shopping, Dijkstra Algorithm, Shopping Robot**

I. Introduction

The massive sizes of shopping complexes with multiple aisles and numerous shelves, it is a task to display each and every product for the consumer to buy. With the additions of new products many older products have to be switched to newer places occupying lesser space. On average, a person living in a city has the presence of at least five different shopping complexes in an approximate range of five miles. Memorizing each and every aisle of the products is a difficult task, and is definitely not practical. To tackle this issue, this research presents an interactive way for the consumer to avoid walking multiple aisles to find the required product and gives the shortest possible route to the given products and then finally to the counter.

To find an optimal path, the best criteria is to analyze and cut short the nodes being explored in the working environment. The best possible way to generate the optimal path is by taking into account the heuristic being used for the path planner, which eventually decides the path to be chosen [1]. Shopping carts have their front wheels freely rotating in but the hind wheels have restricted motion for forward and backward. Keeping in mind this design, [2] developed an interactive Shopping Trolley, which self aligns itself to the walls, or able to turn on its position to load heavy goods thus pointing out the Holonomic driving constraints. Shopping complexes are vast with multiple shoppers in the same aisles and numerous sopping carts being maneuvered at any given instance. The environment is thus changing continuously and to tackle this, an algorithm used in this research [3] is the best fit approach.

Identify applicable funding agency here. If none, delete this.

The intensive study for the controller combined with the D* Algorithm in [4] gives a great insight for the practical implementation of any wheeled robot, in an indoor environment. The rapid random tree method in [5] gives the results for high obstacle avoidance, shorter search time and stronger real time applications which can be used to improve giving better results. This research [6] for shopping cart specifically for elderly people gives a neat result of building a model and then testing it in real time on a smaller scale and aiming to be tested on a bigger grocery store with multiple robots paired up and connected to form a network.

On average the exploration by a shopping cart and the path taken given in [7], points out the number of repeated nodes explored by the same cart and the wastage of efforts in finding the correct aisle, which when used with the help of an algorithm could have been reduced by a significant number. Any cart can be integrated with multiple sensors for perfect feedback control and inter pairing of more than one cart for smooth navigation making the shopping smart [8]. The improved Dijkstra method and the optimal path finder in this research [9] points out to the innovative use to first explore the path with Dijkstra and then applying that path onto the graphical representation to find the optimal path.

The A* algorithm used in [10] has been really helpful in achieving the results of this research and how heuristic shows an important role in node exploration in any path finding algorithm.

II. Methodology

To find the optimal path to all the objects the following pipeline is used:

A. Data preparation

A data-set has been created that has the location for all the items in the store in such a way that the location points are just in front of the item in the aisle i.e. from where the item is easily collectable, along with that, for the corresponding item, the aisle number for the same is stored. The entry and exit point of the aisle for the respective product is stored which will guide the cart to smoothly travel in the aisle.

Next, the user must give the input for the items to buy from the list of items. Based on the selected item, the program extracts the location points and the aisle number where the item is placed.

Here in Figure 1: the brown patch represents the aisle, the green box represent the items to be picked up. The transparent circle represents the end nodes robot can reach which is the starting/exit points of the aisle and the transparent rectangle is the cart.

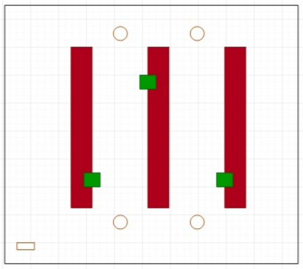

Fig. 1. Workspace

B. Planning Algorithm - A*

On this map, the implementation of A* path planning algorithm is carried out. With the location points and the aisle the program will extract the endpoints of all the aisles. Now the distance between each item location of the item and the endpoints of the aisle in which item is located is calculated using Euclidean distance with the help of the formula:

$$d = \sqrt{(x_s - x_g)^2 + (y_s - y_g)^2} \quad (1)$$

The shortest distance between each item and the chosen aisle's endpoint is picked to calculate the total cost to go from the robot's current position. The calculated heuristic for each aisle and thus the shortest cost is to be the first aisle that the cart should visit.

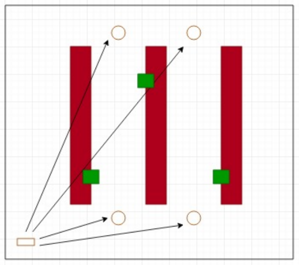

Fig. 2. Check endpoint of the aisle

After selecting the aisle, now the cost to go for each item is calculated as (distance_of_item_to_the_closest_endpoint + distance_to_that_endpoint). in order to select which endpoint robot will travel in order to go through the aisle.

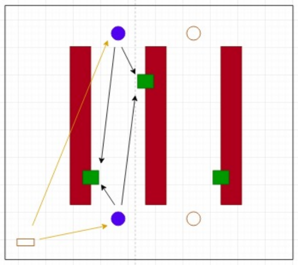

Fig. 3. Check distance between item location and endpoint

After selecting the item and its respective end point of the aisle, the conventional A* algorithm is implemented to generate the optimal path to that respective endpoint i.e. the endpoint is chosen as the goal point for the algorithm [11]. At the endpoint the robot will reorient in such a way that it will be facing the aisle so that it travels in a straight line inside the aisle, meanwhile stopping at all the item locations in that particular aisle and reach to the exit point.

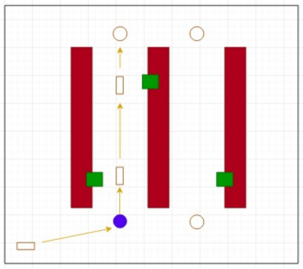

Fig. 4. Path of robot travelling through an aisle

After picking all the items from one aisle the cart stopped at the endpoint and will start the process of node finding again for the next aisle where the next item is located i.e. calculate cost to go first for the endpoint of the aisle then the item in the aisle from the aisle's endpoint.

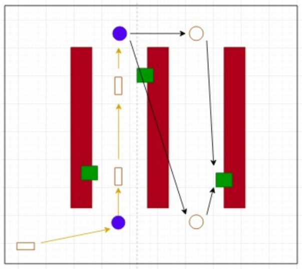

Fig. 5. Check heuristic for the next aisle

Following this, A* path planning to the endpoint of the lowest heuristic of the next aisle to be visited is implemented. The robot re-orients itself again facing the aisle and stops at all the item location before reaching the endpoint of that aisle. Once all the item locations are visited then finally A* is implemented once again to generate the path to reach the end location which will be the cashier's desk from the current position.

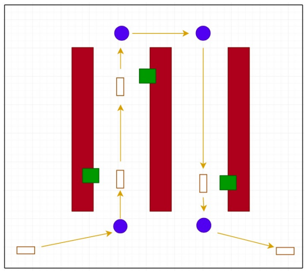

Fig. 6. Path generated using A* algorithm

```
$Q.Insert(x_I)$ and mark $x_I$ as visited
while $Q$ not empty do
    $x \leftarrow Q.GetFirst()$
    if $x \in X_G$
        return SUCCESS
    forall $u \in U(x)$
        $x' \leftarrow f(x,u)$
        if $x'$ not visited
            Mark $x'$ as visited
            $Q.Insert(x')$
        else
            Resolve duplicate $x'$
return FAILURE
```

Fig. 7. A* pseudo code [12]

Fig 7 displays the Pseudo Code for A* algorithm which has been referenced from the research article [12] and displays the nodes to be explored and categorizes them in the visited or the not visited list. If duplicate nodes are being explored, the heuristics plays the role negates the one with greater cost.

*C. Planning Algorithm - D**

As we know the environment of shopping complex is highly uncertain i.e. there can be ambiguous obstacles when the robot is in operation in the work space. D* algorithm can be applied for this ambiguous obstacle when the robot is not inside the aisle.

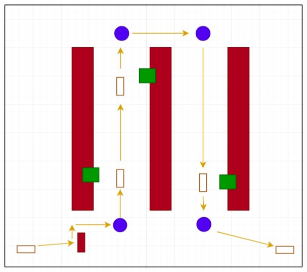

Fig. 8. Path to be generated using D* algorithm

To implement the algorithm in between aisles to avoid/skip already present carts, the algorithm can be updated in such a way that the cart taking higher time (pick multiple products) can be sent into the aisle after a couple of carts are sent which only need one product from the cart.

Sophisticated D* algorithm can be implemented at the same time, which will avoid the cart by taking a curved turn from the standing cart to increase the productivity of the complete setup.

The flowchart showcases the outline of the path planning algorithm chosen for better understanding of the complete process.

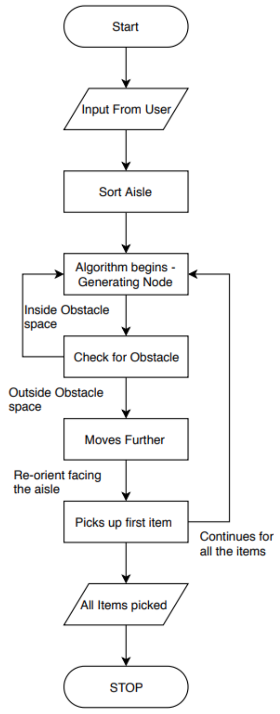

Fig. 9. Methodology flowchart

III. INNOVATION

A. Innovation With respect to planning algorithm

In order to advance the efficiency and accuracy of path planning, in this paper, the D* along with A* algorithm is further optimized. As mentioned in the introduction section we see that this idea of a shopping cart has been usually applied for easy checkout or for elders so they don't need to carry the cart.

The novel approach used for this research is that A* algorithm being applied for all the non-dynamic systems and D* taking over in case of a dynamic system. When it reaches the aisle, node exploration has been decreased and it thus re-orients itself and explores in the direction of the product by changing x or y according to the location.

Thus the exploration needs to be only done to reach aisle and not to move in-between the aisle. Accompanying this, the research takes into consideration the global work space and the local work space. The global work space performs during the non-dynamic system and is therefore practiced when A* is implemented. The local work space plays a pivotal role while executing D* due to the dynamic environment.

Earlier work includes exploration using RRT and also in between aisle which is not necessary if we orient the cart in the direction of the object to be picked up.

B. Innovation With respect to understand the dynamic system

Multiple researches already conducted and as the references represent that the dynamic system is detected using Ultrasonic or Infrared sensor.

The aim is to intend the cart getting data from these key sensors, ultrasonic synced with IR along with a camera module. This will give a better understanding of the dynamic environment and help avoid obstacles when continuous feed-back is present.

IV. Results and Discussions

A. Analytical Results

The two main algorithm, Dijkstra and A*, stated in this research along with the application of Dynamic A* when an obstacle is present can be summarized as follows stating the major differences.

1) STEP 1: Computing Cost
   a) Dijkstra - The cost involved for Dijkstra [11] is the exploration cost to reach the goal, without consideration of any heuristics. This means, the node exploration is in all directions from the starting node.
   b) A* - The cost calculation in A* consists of calculating the heuristic to reach the goal node, along with the similar calculation involved in Dijkstra. Considering both the costs together results in the node exploration leading towards the goal node, thus reducing the node exploration significantly.
   c) D* - This algorithm involves the calculation of cost to come and cost to go, with the involvement of a key component which is the addition of the minimum new cost to go and old cost to goal with the heuristic cost. The important criteria to be kept in mind is that the exploration takes place from the goal node to the start node.
2) STEP 2: Initialization - The initial cost is set as infinity for the goal node, as it will be different for different scenarios. This cost is set as infinity for all the three cases.
3) STEP 3: Update Nodes - The nodes are updated considering the heuristic cost for the latter two and the adjacent node to be explored for Dijkstra.
4) STEP 4: Compute Shortest Path - This involves the mapping of all the cost and nodes explored to get the final path. This process is same for all the three.
5) STEP 5: Main Loop - The loop will continuously run until the goal node is the node that is being explored for further path exploration, which will then stop the loop and the result would be the path.

Thus from the analytical results we see that according to the nodes generation factor, both D* and A* will give better results, one being used in dynamic environment and other in non-dynamic. It is also noted that Dijkstra takes maximum amount of time for computation and A* being takes significantly less due to the cost factor. We can further prove our analytical understanding in the experimental results below.

B. Experimental Results

The total number of nodes explored to travel in between starting point to the aisle endpoint and in between the aisles using Dijkstra path planning algorithm is 79982. The plot of the same as shown below tells that the area in red is the nodes being explored and the one in blue is the path taken by the robot. We here notice that almost 50% of the whole work-space, even when we are not considering the areas between the aisle. If that would have been calculated, the nodes would increase to a very large value.

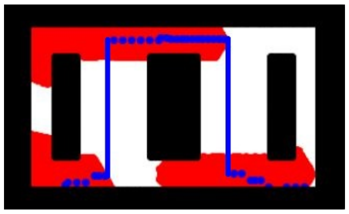

Fig. 10. Dijkstra Output

As shown in figure 10, there is no node exploration in between the aisles because the cart will directly go from the starting of the aisle to the end of it, stopping wherever the product has to be added into the cart. There are two reasons to avoid the node exploration in between the aisles for this application.

1) The nodes to be generated would significantly increase in between the aisles, making the algorithm complicated and will result in more time to compute the same straight path.
2) There could be damage done to the aisles or specifically the products in the aisle if the cart generates a path touching through or very close to the products. The goal for this research is for the cart to enter the aisle, stop wherever the products are to be picked up and then move on to the end of the aisle in a straight path, to jump on to the next one. The path planning is thus done, when crossing in between two aisles, skipping the aisles not required to go into or the area outside the aisles.

In contrast to this, in A* the number of nodes explored are 896, which is significantly less than the Dijkstra algorithm.It is approx 1.1% of the nodes explored by Dijkstra and .5 % of the total work space nodes. Although both of the planning algorithm give the optimal path from one point to another but because of heuristic involved the nodes explored are less in A* hence its execution time is also very less when compared to Dijkstra.

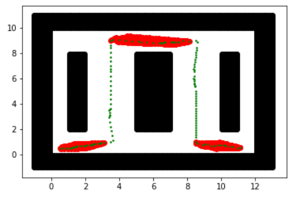

Fig. 11. A* output

The nodes as noticed in fig 11, displays that the reduced area of exploration is present when A* algorithm is applied and the computation time for the same is increased many folds. The path to be travelled by the cart is found quickly for the customer to start shopping.

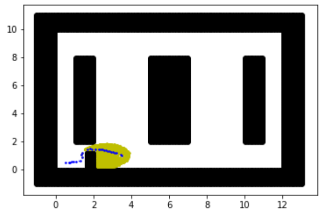

Fig. 12. D* Output

If there is any presence of an obstacle in the environment which is quite obvious, considering the work space, the A* algorithm is switched to D* to avoid the obstacle and as noticed in the figure, there is a new node exploration being conducted to reach the goal node, in a shorter area, and not the complete work space.

Thus, Figure 12 depicts the re-planning done to achieve the optimal path using D* in an dynamic environment. If an obstacle comes in the work-space after the path has been planned, re-planning comes into play from the goal node now being the start point. The path seen is no longer the optimal path like the A*. Also the curves of this path can further be made smoother by using Dublin's path, or B-spline. Here it is seen that less that 50% of the path is being re-planned avoiding the obstacle even more easily.

Parameters	Dijkstra	A*	D*
Number of nodes	79982	896	500 more than Non-Dynamic
Computation time	17 minutes	3 minutes	5 minutes
State space exploration	Approx. 50% of the workspace	Less than 1% of the workspace	Less than 1.5% of the workspace

The results shown in the tabular column showcases the comparison between Dijkstra and A* algorithm followed by application of D* for a certain region.

As noticed, there is a significant decrease in the number of nodes being explored for the second algorithm compared to the first one, and if the second algorithm is applied with dynamic obstacles, the outputs, although slow, is optimal.

V. Conclusion

The results from the nodes explored by using the algorithm stated above and the dynamic vicinity of a shopping complex, the objective of this research was to reduce the time for shopping in any shopping complex. The two approaches being showcased in this report are Dijkstra and A* (in combination with D* for dynamic obstacles). The shopping complexes focused in particular are the grocery stores with multiple aisles forcing the user to traverse on each and every aisle to find a particular product. The dynamic A* being used as the method of implementation gives the best results in a optimal time.

Time, energy, cost of each and every shopper is saved and specifically the health of elders is kept in mind as the shopping cart navigates on its own, through the aisles with the optimized path. Dijkstra, which doesn't have any cost function involved, checks on each and every node being explored in the vicinity of the nodes explored, thus giving us about 80000 node exploration in contrast to A* algorithm which can follow a better optimized path in about 1000 nodes. As stated above and shown in the graphical representation, if there is a presence of any obstacle, which is quite ordinary in any complex, the cart can re-plan its path with the help of D* algorithm.

Once achieved on a smaller scale, this research can be scaled up to an extent when all the carts will follow symmetrical paths along the aisles and while changing aisles to avoid collisions and thus implementing this methodology successfully.

VI. Future Work

A. Robotic Arm

Introduction of the robotic arm and a barcode scanner to pick, scan and release the item in the container to make the system more convenient and have faster checkouts. Thus humans do not need to go around the mall, therefore saving ample amount of time.

B. Frontier Exploration with SLAM

Having multiple carts connected to each other to have a better perception of the environment i.e. rapidly updating obstacle space for all the carts. This can be done by having frontier exploration and SLAM. This will ensure no collision along with the reduction in cost to generate the initial map

C. Mobile Application

A mobile application (android/ios) can be used to show the customer where the cart is and the map to pick up all the object. This will additionally be essential in feeding the input of items to be picked and to keep a tally of which all are picked and which all are left to be picked up.

D. Increase the actions of movements

Increasing the number of actions i.e. introducing the backward motion. This will be immensely helpful when trying to avoid obstacles between the aisle since it can go back and other actions to avoid it can be taken. If this does not take place it will have to stand for the obstacle to move inside the aisle thus the benefits decreases.

References

[1] David Fergusan, Maxim Likhachev, Anthony (Tony) Stentz, "A Guide to Heuristic-based Path Planning," International Conference on Automated Planning and Scheduling, June, 2005.

[2] Michael Goller, Thilo Kerscher, Johann Marius Zollner, and Rudiger Dillmann, "Obstacle Handling of the Holonomic-driven Interactive Behavior-operated Shopping Trolley InBOT," Robot Motion and Control 2009, LNCIS 396, pp. 463 - 472

[3] Khalid Al-Mutib, Mansour AlSulaiman, Ebrahim Mattar, "D* Lite Based Real-Time Multi-Agent Path Planning in Dynamic Environments," Third International Conference on Computation Intelligence, Modelling and Simulation, 20-22 Sept, 2011 pp. 170 - 174

[4] I-Hsum Li, Yi-Hsing Chien, Wei-Yen Wag, Yi-Fend Kao, "Hybrid Intelligent Algorithm for Indoor Path Planning and Trajectory-Tracking Contol of Wheeled Mobile Robot," International Journal of Fuzzy Systems, Vol 18, No. 4, August 2016, pp. 595608

[5] Yi Cheng, Yun Wang, "Research on path planning of intelligent shopping cart in large shopping mall ," The 30th Chinese Control and Decision Conference (2018 CCDC), pp. 26582661

[6] Jorge Sales, Jose V. Mart, Ral Marn, Enric Cervera, and Pedro J. Sanz, "CompaRob: The Shopping Cart Assistance Robot," International Journal of Distributed Sensor Networks, Volume 2016, Article ID 4781280, 15 pages

[7] Jeffrey S. Larson, Eric T. Bradlow, Peter S. Fader, "An exploratory look at supermarket shopping paths," Intern. J. of Research in Marketing 22 (2005), pp 395 - 414

[8] Mr.P. Chandrasekar, Ms.T. Sangeetha, "Smart Shopping Cart with Automatic Billing System through RFID and ZigBee," ICICES2014, ISBN No.978-1-4799-3834-6/14

[9] Hwan Il Kang, Byunghee Lee, Kabil Kim, "Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm," 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, DOI 10.1109/PACIIA.2008.376, pp. 1002 - 1004

[10] Akshay Kumar Guruji, Himansh Agarwal, D. K. Parsediya, "Time-Efficient A* Algorithm for Robot Path Planning,“ ICIAME 2016, Procedia Technology 23 (2016), pp 144 - 149

[11] Peter E. Hart, Nils J. Nilsson, Bertram Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Transactions of systems science and cybernetics, Volume: 4, Issue: 2, July 1968, pp 100 - 107

[12] Steven M. LaValle, "Planning Algorithms," Cambridge University Press