ROBOCART: A self-navigating shopping cart for the supermarket

Cesar Alan Contreras, Runlong YANG, Yang LIU, Yongle HUANG, Zongyuan ZHANG
School of Computer Science
University of Birmingham

***Abstract*—The ROBOCART (tracker) is a fully-automatic supermarket cart designed to free customers hand while shopping. Our system has the function of human-tracking, trajectory prediction, automatic path-planning, automatic charging. In this article, we compare our planned path from A* algorithm with the most optimal path and explain our design choice. We also illustrate how we made our customer-following task smarter by introducing Kalman-Filter into our system. And we prove that it successfully saves up to 13% of the time while following a customer in certain scenarios with Kalman-Filter trajectory predictions.**

I. Introduction

Nowadays AI has become more and more popular and normal in most domains. Shopping is a necessity in daily life, and AI technology will make it easier and more convenient. According to the Amazon AWS official blog [1], Back in 2017, Amazon developed its first self-serve supermarket called Amazon GO. The goal of the developers is to achieve the 'no lines, no check out', which means there is no queue required for the customers any more. The service is provided via the mobile app that will be applied to scan the barcode of products and then check out by itself. On the one hand, supermarkets like Amazon GO bring great convenience to the customers, but also there are some obvious limitations that it is not friendly to the special groups such as the seniors and people with disabilities. Not only that, this emerging technology will bring a big impact to traditional supermarkets.

Because of this, our team intends to develop a supermarket intelligent shopping robot, which is able to follow the user and show the optimal path to them and to their predicted positions. At the end of the shopping, the robot can be able to find its way into a particular area alone, paying for the items that the user chose, or going back to the energy station to recharge its battery in time.

Compared with the traditional or general shopping method, shopping under the service of a supermarket robot is high-efficiency as the robot can guide the user to the location of items the user wants, whats more, the customers do not need to wait in a queue since the payment will be implemented by the robot directly. In case some customers include seniors and disabilities, the robot can help them to approach the items they want easily, which can provide them with a better shopping experience. On the other hand, the supermarket robot is different from the Amazon GO. The advantages of the former one is that it has a better integration with the traditional shopping methods and provides the users more convenience with less impact.

II. Related Work

A. Path Planning

In general, three categories of algorithms are applied to robot path-planning problems by Cais research [2]. The first category is search-based algorithms, such as the Dijkstra and A* algorithms [3]. The second category is sampling-based algorithms, such as the PRM and RRT algorithms [4]. The third category is intelligence-based algorithms, such as genetic algorithms [5] and ant colony algorithms [6]. Of these three categories, intelligence-based algorithms are more commonly used in an unknown or partially known environment mobile robot problems, and less commonly used in given environment mobile robot problems due to their high time complexity [7]. The remaining two categories of algorithms are very different in principle, the most obvious difference is that they can handle different kinds of maps. The search-based algorithm requires a grid map or landmark map, whereas the sampling-based algorithm can be applied to arbitrary, irregular maps due to its nature of randomly sampling points on a map.

The Dijkstra algorithm and the A* algorithms are the most commonly used in search-based path planning. Compared to the Dijkstra algorithm, the A* algorithm adds a heuristic function based on a priori information about the start and end points to the breadth-first search strategy. This allows the A* algorithm to reduce the number of nodes in the search extension and increase the efficiency of the path search. For the known conditions of our problem, the robot position and the target point are known, which means the prior information of the A* algorithm is known. There for, the A* algorithm is a more efficient path-planning algorithm for this task.

Another task is to avoid crowds in the supermarket during path planning. In the A* algorithm, crowds can be set as part of a grid of obstacles and the map can be updated in real-time to achieve dynamic path planning. The problem is that crowds are smaller obstacles than shelves, and if they occupy the same size grid as the shelves, the robot's path will be blocked. However, if the grid size is reduced to the same size as a person, the number of grids increases, the computational speed decreases and the real-time performance is reduced. A solution would be to have smaller grids at crowds and larger grids at the rest of the open space and where there are large obstacles.

B. Real-time tracking with cameras

In the field of robot object tracking research, the following types of tasks can be classified: Single-Object Tracking, Multi-Object Tracking, Person re-identification (Re-ID), Multi-target Multi-camera Tracking (MTMCT), etc. For the supermarket robot, we want to implement the basic tracking is real-time single-object tracking. We, therefore, focus in this section on reviewing the existing approaches for real-time single-object tracking with cameras.

Single object tracking has undergone rapid development in the field of scientific research and can be divided into generative and discriminatory methods. The generative method starts by building an object model or extracting objective features, and then searches for similar features in subsequent frames, iterating step by step to achieve object localization, e.g., Kalman filtering and mean-shift [8]. As for the discriminatory method, it considers both the object model and the background information and extracts the object model by comparing the difference between the object model and the background information to obtain the position of the object in the current frame, e.g., Struck [9] and TLD [10].

In recent years, with the rapid development of technology in the field of object tracking, many motion models have been introduced into the object tracking framework with relatively good results, and the Kalman filter algorithm, as one of the best, is commonly used in linear Gaussian distribution scenarios [11]. To solve the object-tracking tasks in different complex scenarios, researchers have designed many object-tracking algorithms based on the idea of multi-feature fusion to enhance the effectiveness of traditional particle filter tracking models. Brasnett et al. propose a multi-featured object tracking method that incorporates colour, texture and edges, and this model provides improved and more stable performance than tracking with a single feature [12]. Spengler et al. exploit an adaptive multi-feature fusion tracking method that uses online adaptive methods to adjust the weighting of different features to obtain better tracking performance [13]. Ma et al. proposed a feature fusion method, using colour contrast similarity and directional consistency as features for object edge detection in colour images for object tracking [14]. Liu et al. adopt a method for adaptively combining object image colour and shape information based on the idea of fuzzy logic to describe the observation model to achieve object tracking to improve the reliability of the model [15]. Sun et al. explore a particle filtering object tracking model based on the fusion of image corner features and colour features, which improves the sampling efficiency of the particles and enhances the robustness of the algorithm [16].

C. Kalman Filter

With the development of technologies such as robotics and autonomous driving, the requirements for the accuracy and stability of prediction effects are becoming increasingly. The Kalman filter, as the optimal method for state estimation, has become more and more popular and has been widely used in various fields. Nowadays, many relevant studies have been published. Iwao et al. have studied the applications of the Kalman to predict traffic volume in a short period. The experiments provided by them show that the Kalman filter prediction system is better than the original detection system with a minimum 9% to no more than 30% error. [17]What is more, Ashraf et al. who represented a framework to estimate the next time pose of moving objectives applied the Kalman filter. In conclusion from that is the Kalman filter has advantages in predicting the pose without waiting for it, great contributions are provided by the algorithm in terms of predicting the future pose of the robot in a dynamic system. [18]Apart from that, Fang investigates a new RACKF which improves the accuracy of the noise statistic estimation by retaining as many noise statistic estimation terms as possible. [19] The applications of the Kalman filter are various, Pei-Hwa et al. have discovered research on the detection and prediction of the tide level in a short-term. The result presents there is a satisfying predictive level obtained via applying the Kalman filter, the algorithm works well both online and on PC. [20]

III. DESCRIPTION

For the elaboration of the project a grid map is created. The grid map is the surface the robot is supposed to navigate through. The project tries to emulate a supermarket with multiple sections in which a human is supposed to be doing his shopping in. The robotic car is then required to follow the human around the map while doing his shopping unless for some reason it requires to recharge its battery. If the situation arises the robot plans a path to battery stations, and then finds its way back to the human.

The implementation requires always knowing the position of the robot, and the energy levels it has during the given time. Other decisions require user input to be able to know what the current mode of operation is. The robot can then either continue following the human, guide the human to a store location, or move to the checkout counter.

IV. ALGORITHMS AND FRAMEWORK

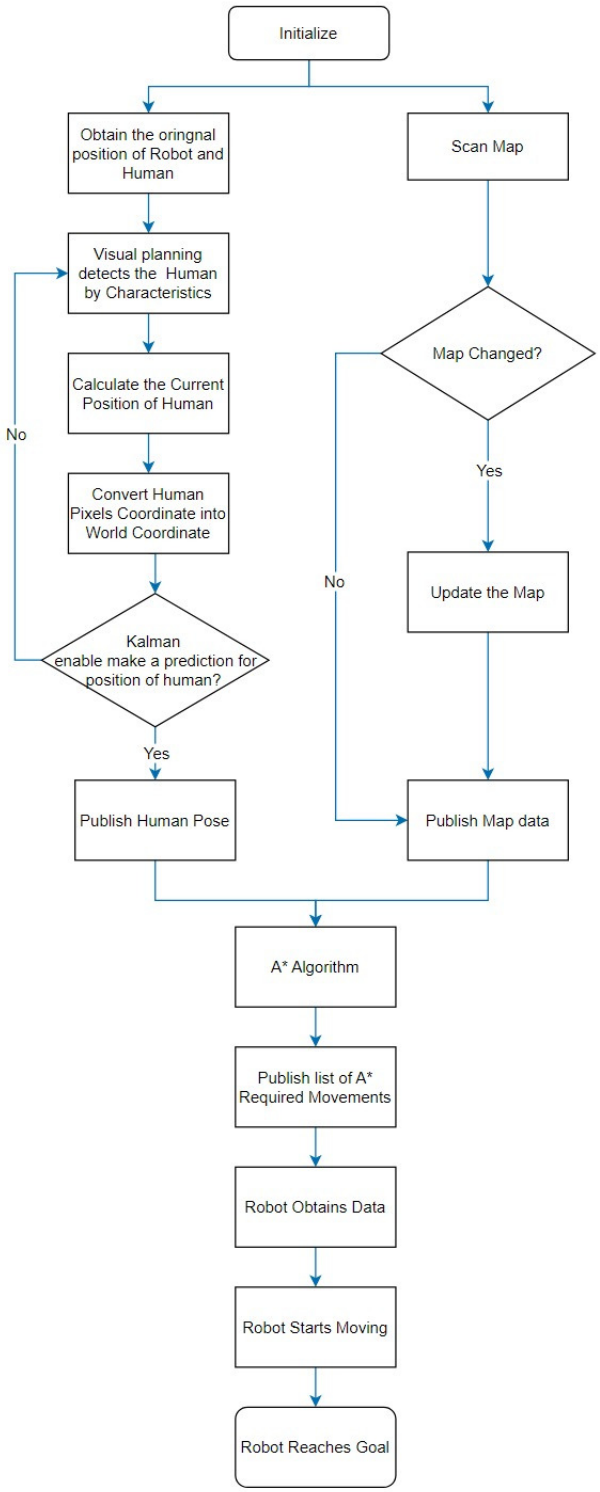

Fig. 1. Flowchart of the project

The figure 1 represents a clear logical process of the whole project. What is notable is that the position of the human is detected by the visual characteristics, the human is simulated as a red rectangular, and the position of it is detected by the camera via OpenCV program. Then the local coordinate of the human will be converted to the world coordinate that is easily applied by the Kalman filter prediction and path planning. Since the map of supermarket can be dynamic sometimes it is necessary to refresh the data of the map in time so that the correct data could be used in A* algorithm to plan the right path for the robot. Then the desired movements are transferred to the robot and it moves to the goal as required.

A. ALGORITHMS

1. Algorithm A*_Path_Planning
2. Initialize open_set and close_set as ∅
3. Add start_state to open_set and set F of start_state as **0** (minimum)
4. If open_set is not ∅, choose the node with the minimum F from open_set:
5. If node n is the final_state:
6. Link the parent nodes from the final_state to the start_state to return the path
7. Else:
8. Remove node ***n*** from open_set and add it to close_set.
9. For m in the adjacent nodes of ***n***:
10. If ***m*** in close_set:
11. Set ***n*** is the parent of ***m***
12. Compute F of ***m***
13. Add m into open_set and sort open_set

Fig. 2. A* Pseudocode

1) A algorithm:* The implementation of A* search algorithm to be able to find the optimal path between two points (current location and objective) while avoiding obstacles, based on grid sizes using data from the bitmap. For the path planning, the robot needs to do two tasks, which are finding the path to the checkout counter and finding the path to the charging station. The known condition is that the general topography of the supermarket, including the position of the shelves, the checkout counters and the charging stations is static and the placement of these things in the supermarket is more regular. The robot simultaneously needs to avoid the crowd, which can be considered as part of the obstacles on the map. Therefore, these tasks could be concluded as finding a path from a known robot position to a known target point on a known, regular map.

Generally, it is necessary to define the grids of the map to applied the A* algorithm. The core cost function about the A* is :

$$F = G + H \tag{1}$$

Where the G is the cost of movement from the start point to the current grid and the H stands for the cost from this specific point to the final state, therefore, the F is the total cost of the grid where it is located. Then the lists called opened and closed will be defined relatively, then the map is seen as grids and some of them have been defined as the obstacles or walls already. In our project, the start point is the location where the robot is , and the final state depends on the mode user chosen. Under the selection mode, the end point of path planning is the customer or human we defined in the map, under the charging mode the goal is the battery station point, if the battery volume is under 30%, the robot will find a path to the battery station, and under the checkout mode the goal should be the payment area.

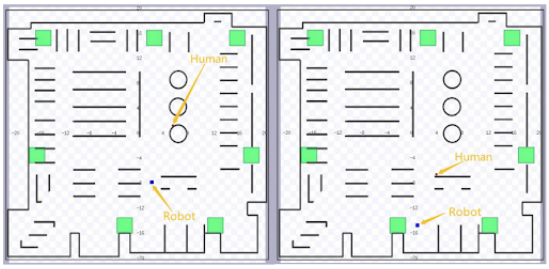

Fig. 3. Failing cases of Path Planning

After these arrangements have already been done well, we can start to find the shortest path between the start point and the goal the user chose. First of all, initialise the defined lists including opened and closed, the grids around the start point and the grid of itself will be added into the opened list and set the start grid S as their parent, what is notable that in our path planning, the around grids are no more than four (top, bottom, left and right), since the robot is defined to move as these four directions; And then, move this start grid into the closed list, which means the grids in closed list will not be checked any more. Thirdly, compute the F of each grid in the opened list and sort it to find the one with minimum F called A, then move A into the closed list. Also add the accessible grids around A into the opened list (ignore the walls and obstacles) then compute the F of these grids and set A as the parent; name the near grid as C, if C is already in the opened list, compute the path from start grid S to C (the path will pass the A), the value of G will be the standard to judge if the parent needs to be updated, if the new value of G is lower than previous one, then the parent grid is A and the value of F should be updated; otherwise nothing will be changed. Finally, keep finding the grid with minimum value of F and move it into the closed list and repeat the previous procedures again until the convergence criteria is met, one is that the goal grid is in the opened list and anther is that there is no grid in the opened list and it means there is no suitable path for the project.

In the majority of cases, the A* algorithm is able to achieve Robot's Path Planning Problems. However, there are still some failing cases where the robot is unable to find the path to the human. As shown in image 3, when the human is tightly close to the obstacles, the robot is unable to reach the human due to the size of the robot itself.

2) Object Tracking: For the tracking a camera tracking algorithm, that uses a discriminatory method comparing current data with background information, extracting the human and other objects positions in the current frame. For the simulation, at the start we consider the data obtained directly from the map and ROS topics.

For tracking targeted humans, in the actual situation, the camera is usually used to scan the detection area to obtain the image. Then according to the relevant characteristics of the human determine whether the human is in it. Once the human is found, its position will be recorded and updated in real-time on each frame to complete the tracking. In our simulation, screen capture is to simulate the camera in the actual situation. Apart from that, the detection area in the simulation is the grid map which is a black-and-white map. In this simulation, the human in the grid map is shown as a red point to represent the humans features, and a set of x and y coordinates is obtained and then transformed into world coordinates. In some cases the map is scanned and if there are changes to it the grid map is updated.

3) Human trajectory prediction with Kalman filter: The process model of the Kalman system is described by a Linear Stochastic Difference Equation:

$$x_k = F \cdot x_{k-1} + B \cdot u_k + \varepsilon_k \tag{2}$$

And the measurement model is:

$$z_k = H \cdot x_k + \delta_k \tag{3}$$

X_k is the system state at time k, and u_k is the control of the system at time k. F is the state-transition matrix that stands for the linear transformation from last state to the next state. B is the Input-control Matrix. In terms of object movement, F and B indicates the underlying motion model of the system. ε_k is the process noise that usually assumed to be drawn from a zero mean multivariate normal distribution with covariance matrix Q_k.

H is the observation (measurement) matrix, and v_k is the observation (measurement) noise which also follows a normal distribution with covariance matrix R_k.

Kalman Filter contains prediction and correction step. In the prediction stage, the filter uses the estimation of the previous state to make the estimation of the current state. To start with, we calculate the best estimate of next state by the algorithm:

$$\hat{x}_k = F \cdot x_{k-1} + B \cdot u_k \tag{4}$$

In our project, our state is the position and moving velocity of the human as shown below:

$$x_k = \{x_k, y_k, v_{x_k}, v_{y_k}\} \tag{5}$$

we mainly focus on the position information of the state, and the velocity vx_k and $vy_k$is calculated using equation below:

$$v = \frac{(x_{k-1} + x_k)}{\Delta t} \tag{6}$$

Δt is the time interval between last state and current state. And if we inserted more than one position and make more than one prediction, the velocity will be the mean value of all previous velocity.

And our state-transition matrix is defined as:

$$F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

with this F matrix, we assume the humans movement is uniform motion in a straight line. Since we are predicting a human movement with no control signal, so there are no B matrix and control vector in our case.

And we update the covariance of our prediction by the following equation:

$$\hat{P}_k = F \cdot P_{k-1} \cdot F^T + Q_k \tag{8}$$

In the correction (updating) step, the filter optimizes the predicted value obtained in the prediction step by using the observed value of the current state to obtain a more accurate new estimated value.

The corrected state estimation is calculated by:

$$x_k = \hat{x_k} + K_k \cdot (z_k - H \cdot \hat{x_k}) \tag{9}$$

The covariance is also updated by:

$$P_k = (I - K_k \cdot H) \cdot \hat{P}_k \tag{10}$$

K_k is the Kalman Gain, it can be computed by:

$$\begin{aligned} K_k &= \hat{P}_k \cdot H^T \cdot S_k^{-1} \\ &= \hat{P}_k \cdot H^T \cdot \left(H \cdot \hat{P}_k \cdot H^T + R_k\right)^{-1} \end{aligned} \tag{11}$$

In our simulated case, the measurement matrix is set as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{12}$$

When we input more points into our predict() function, we take the new point as a measurement and use it to reduce our noise covariance. So when we insert more points (more human position) into the function, the next prediction will be more accurate. (as shown below)

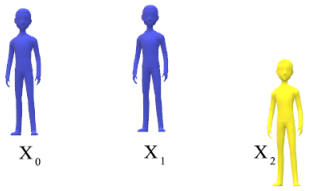

Fig. 4. X_2 prediction based on two inserted state (X_0 and X_1).

The yellow human above is our predicted position when we inserted previous human trajectory points into our algorithm. In this graph above, X_0 and X_1 are the trajectory points we inserted and the X_2 is the predicted future position.

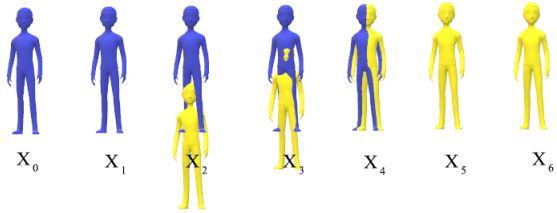

Fig. 5. X_5andX_6 prediction based on more inserted states

From the graph above, we can see that the predicted human position X_5 and X_6 is more accurate than the previous X_3 and X_4 prediction. The accuracy of prediction increased when more states were inserted into our Kalman filter algorithm.

However, when we make further prediction based on previous future prediction(for example, predicting next position based on the yellow X_6 above), the uncertainty will accumulate because the covariance of the estimation error will increase.

V. EXPERIMENTAL RESULTS

In terms of the A* we applied in the project, in order to reduce the computational power, we decided to compute the cost value F step by step. The reason for it is to decrease the source requirement to acquire a great computational speed, on the contrast, the system will lose the shortest path the figure below represents the difference between the shortest path and the real path from our A* algorithm:

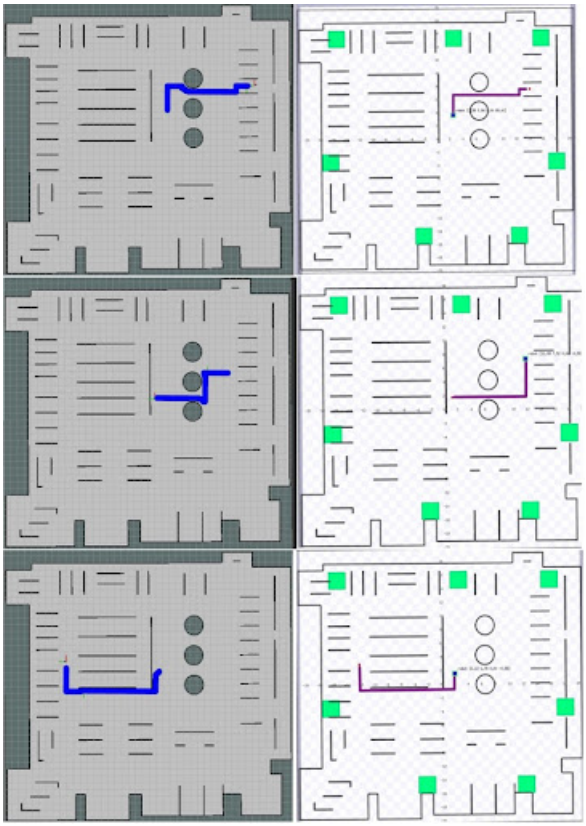

Fig. 6. Path comparison

The left part of figure 6 presenting the path that A* algorithm in our project from three groups start point and the goal, it is easy to find that the robot tends to follow the path where it is near the obstacles (walls). This is because every step the algorithm we applied only computes the cost of grids around the current state grid, until there are obstacles in the surrounding grids. The advantage for the method we applied is it could save much computational resources in the simulation, in the other words, the path will be generated fastly. However, during the real world A* algorithm path planning, the computational requirements could be satisfied all the time, and there is no need to desire the run time, on

the contrary, the goal is to acquire the optimal path between the start point and the end. Since the implementation of the project relies on the simulation, it cannot be ignored that the computational speed is an important parameter in the work.

A. Kalman filter-based trajectory prediction in path planning

Due to the complexity of the supermarket environment and the performance limitations of the robot itself, special situations can easily arise when the robot is following a human customer. Some examples are: the distance between the customer and the robot becomes too far due to the customer moving faster than the robot, an obstacle in the robot's path makes it impossible to follow the customer, the robot's power level is too low and a new robot needs to be re-dispatched from another location, etc. In these cases, the robot needs to carry out optimized path planning so that it can reapproach the customer's location in the shortest possible time. Obviously, as the customer is always moving, this is a path-planning problem with a moving object as the target. A possible solution to this problem is to use the Kalman filter to perform a prediction of the customer's trajectory and use the prediction results in the A* path planning algorithm to derive an optimized path for the robot to follow.

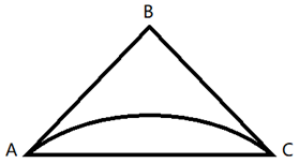

Fig. 7. Trajectory prediction diagram

Consider a simple example, assuming the following figure 7, where the robot's initial position is at point A and the human customer's initial position is at point B. The human customer will walk from point B to point C, and the robot will move at a similar speed to that of the human customer. Firstly, if the robot adopts the strategy of directly following the customer's walking route, the robot will move from point A to point B and then from point B to point C. The distance travelled by the robot is the sum of AB and BC. Secondly, without the use of trajectory prediction, the path planning algorithm plans the path from the robot's current position to the customer's current position in real-time, and the robot's trajectory will be similar to the arc from point A to point C in the diagram, and the robot's moving distance will be the arc length from point A to point C. If the Kalman filter-based trajectory prediction is used, the path planning algorithm can directly plan a path from point A to point C. The travel length AC is obviously shorter than the sum of AB and BC or the arc length between points A and C, allowing the robot to get closer to the human customer faster. This shows the clear advantage of using the Kalman filter-based conjunction prediction in path-planning algorithm in certain situations.

B. Experiments in trajectory prediction application

1) Brief description of the experiment: During the simulation, the following examples demonstrate the superiority of using the Kalman filter-based conjunction prediction with the A* algorithm in the supermarket path planning robot problem. This example is representative of a class of problems where the robot has a closer route to the predicted future position of the customer than the length of the route if he was to follow the customer's route, a type of problem that often arises in supermarket path planning problems. In the map which is used in this example, as shown in figure 8, the three horizontal lines on the left represent supermarket shelves and the circles on the right represent the column in the supermarket. The human customer and the robot moves at a fixed speed, with the robot moving slightly slower than the customer. The Kalman filter used in this example has a prediction step of one step and predicts the customer's position after two seconds.

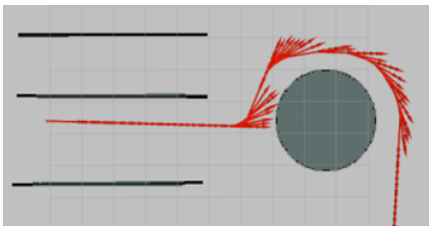

Fig. 8. Planned route for customers

This example is a scene reenactment of a human customer going to walk around the columns, as shown in figure 8, where the red arrows represent the customer's planned walking route, coming out from between the second and third shelf and preparing to walk around the column from above towards the bottom of the map. Figure 9 shows the state of the robot as it walks to the shelf exit (represented by the blue square on the left of figure 9. The smaller blue circle at the top right is the customer's true position at this moment, and the larger circle below is the customer's next moment position predicted based on the Kalman filter. Figure 10 shows the robot's path plan at this moment without the trajectory prediction, which would make the robot follow the customer around the column from above. Figure 11 shows the robot path plan at this moment with the trajectory prediction, which would make the robot take a shorter route directly going under the column to the customer's next moment predicted location. Obviously, as the customer will continue walking down as planned, path planning using trajectory prediction will result in shorter distances and greater efficiency.

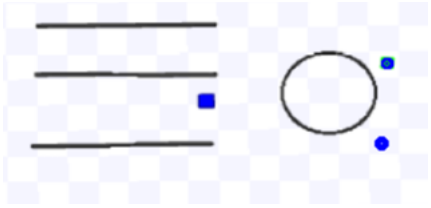

Fig. 9. Position of robot and customer at a certain moment

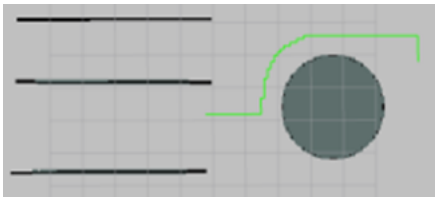

Fig. 10. Robot path plan without trajectory prediction

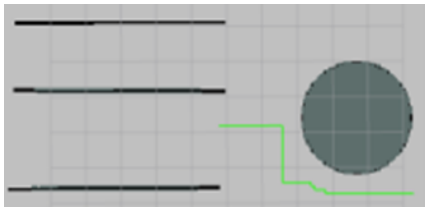

Fig. 11. Robot path plan with trajectory prediction

2) Comparative trial with trajectory prediction: In the supermarket path planning problem, the expected result is that the robot can reach the customer's location in the shortest time. A comparative trial of whether the robot can reduce the time cost with the use of Kalman filter-based trajectory prediction in this walk around the columns example can demonstrate the performance of using the trajectory prediction in the supermarket path planning problem.

TABLE I
TIME COST OF TWO GROUPS

Time cost of experimental group /s	Time cost of control group /s
49.34	53.92
46.62	58.37
51.02	60.70
59.76	67.87
52.94	57.87
51.25	62.76
52.78	59.56
49.72	63.54
51.36	57.55
56.66	59.44

The control group for the comparison trail was path planning performed when trajectory prediction was not taken, and the experimental group was path planning performed when trajectory prediction was taken. The final time cost for the robot to reach the customer's location in these two experimental groups is shown in table I.

As can be seen in table I, the arithmetic mean of the time cost of the control group is 60.16 seconds, and the arithmetic mean of the time cost of the experimental group is 52.15 seconds. This represents a 13% reduction in time cost with trajectory prediction compared to without it. Figure 12 shows the overall trends in time cost for two groups, indicating that the use of trajectory prediction can reduce time cost obviously. In summary, the comparative trial proves that trajectory prediction has a good performance in the supermarket path planning problem.

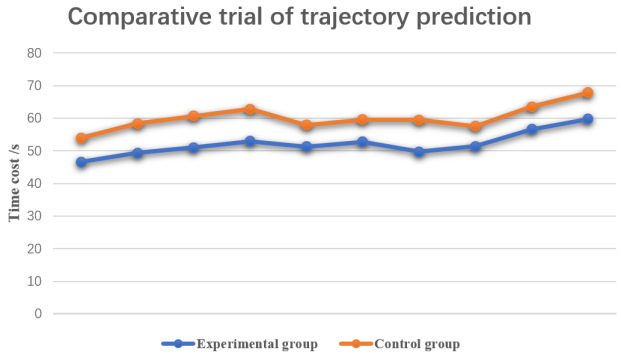

Fig. 12. Comparative trail of trajectory prediction graph

VI. CONCLUSIONS

This project designs the software part of a shopping cart robot for supermarkets that automatically follows customers and returns to a charging station for recharging when the battery is too low. The program updates the map and locates the customer using computer vision. The program plans the path from the robot to the customer using the A* algorithm and Kalman filter-based trajectory prediction on the customer to let the robot follow the customer.

The project is validated using simulations and can achieve its objectives in a variety of situations. The advantage of the project is that it uses less computational power to plan shorter path during path planning, attribute to the use of the predicted location of the customer as the planning target and the use of the locally optimal A* algorithm for path planning, proving that for various cases path planning into the future might save movement time and create better paths. The disadvantage of the project is that the robot's motion is limited to only two orthogonal directions, resulting in limited flexibility. This is because the robot's next position when using grids for path planning only considers the four immediately adjacent grids, which can be solved by increasing the search for the next position to eight surrounding grids.

VII. REFERENCES

REFERENCES

[1] AWS team, Amazon Go Unmanned Retail Store Revealed, April 2021, https://aws.amazon.com/cn/blogs/china/amazon-go-unveils-unmanned-retail-store/

[2] C. W. J. C. S. S. Kuanqi Cai, "Mobile Robot Path Planning in Dynamic Environments: A Survey," arXiv preprint, p. 2006.14195, 25 6 2020.

[3] N. J. N. A. B. R. PETER E. HART, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE TRANSACTIONS OF SYSTEMS SCIENCE AND CYBERNETICS, pp. 100-107, 1968.

[4] S. M. LaValle, " Rapidly-exploring random trees: A new tool for path planning," Report No. TR 98-11, 1998.

[5] J. T. a. S. X. Yang, "Genetic Algorithm Based Path Planning for a Mobile Robot," lEEE lnternational Conference on Robotics & Automation, pp. 1221-1226, 2003.

[6] C. L. C. a. C.-C. C. Ying-Tung Hsiao, ”Ant Colony Optimization for Best Path Planning,” lntematbnalSymposium on Communications and Information Technobgies, pp. 109-113, 2004.
[7] P. V., Path Planning for Vehicles Operating in Uncertain 2D Environments, Butterworth-Heinemann, 2017.
[8] Comaniciu, D., Ramesh, V. and Meer, P. (2000) Real-time tracking of non-rigid objects using mean shift. In Conf. Comput. Vis. Pattern Recognit. (CVPR). 2000. IEEE. pp. 142149 vol.2.
[9] Hare, S., Golodetz, S., Saffari, A., et al. (2016) Struck: Structured Output Tracking with Kernels. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38 (10): 20962109. doi:10.1109/TPAMI.2015.2509974.
[10] Kalal, Z. (2012) Tracking-learning-detection. IEEE Transactions on Pattern Analysis & Machine Intelligence, 34 (7): 14091409.
[11] Fu, J. and Xu, C. (2019) A survey of single object tracking methods. Nanjing Xinxi Gongcheng Daxue Xuebao, 11 (6): 638650.
[12] Brasnett, P., Mihaylova, L., Bull, D., et al. (2007) Sequential Monte Carlo tracking by fusing multiple cues in video sequences. Image and vision computing, 25 (8): 12171227.
[13] Spengler, M. and Schiele, B. (2003) Towards robust multi-cue integration for visual tracking. Machine vision and applications, 14 (1): 5058.
[14] Ma, Y., Gu, X. and Wang, Y. (2009) Feature fusion method for edge detection of color images*. Journal of Systems Engineering and Electronics, 20 (2): 394399.
[15] Liu, Y., Wang, B., He, W., et al. (2006) Fundamental Principles and Applications of Particle Filters.
[16] SUN, W., GUO, B., ZHU, J., et al. (2010) Robust Object Tracking via Hierarchical Particle Filter. ACTA PHOTONICA SINICA, 39 (5): 945950. doi:10.3788/gzxb20103905.0945.
[17] Iwao Okutani and Y. J. Stephanedes, Dynamic prediction of traffic volume through Kalman filtering theory, Transportation Research Part B: Methodological, vol. 18, Art. no. 1, 1984, doi: https://doi.org/10.1016/0191-2615(84)90002-X.
[18] A. Elnagar, Prediction of moving objects in dynamic environments using Kalman filters, 2001, pp. 414419. doi: 10.1109/CIRA.2001.1013236.
[19] X. Fang, Robust adaptive cubature Kalman filter for tracking manoeuvring target by wireless sensor network under noisy environment., 2022, doi: 10.1049/rsn2.12331
[20] P.-H. Yen, C.-D. Jan, Y.-P. Lee, and H.-F. Lee, Application of kalman filter to short-term tide level prediction, Journal of Waterway, Port, Coastal, and Ocean Engineering, vol. 122, Art. no. 5, 1996, doi: 10.1061/(ASCE)0733-950X(1996)122:5(226).

VIII. CODE

https://github.com/Cesar514/Final-Project