Car First or Pedestrian First? Motion Prediction and Planning in Human-Robot Interactions

Xu Lian(✉)

Shenzhen Middle School, Shenzhen, China
lancelian8086@yeah.net

Abstract. As the demand for autonomous agents such as intelligent vehicles and robots' increases, effective and safe human-machine interaction must be ensured. Therefore, machines should be able to anticipate the motions of humans and solve appropriate schemes to achieve their goals. To accomplish this, a sequence of future positions must be generated according to some historical positions extracted from videos by the predictive neuron network, and the planning algorithm has to find and execute the best scheme in a limited window. In the paper, we propose to use a long-short-term-memory (LSTM) network for prediction and an A* algorithm to solve the planning problem. With these methods, we successfully combined these two parts and conducted simulations where autonomous cars have to reach targets efficiently while avoiding pedestrians in a safe manner. Besides, we compared our predictive models with other models on the UCY data-set. We also conducted experiments to compare the resulting paths with our prediction and those with naive and simple linear prediction. The result showed that our system can effectively generate safe and efficient future paths for autonomous vehicles.

Keywords: Human-robot interaction · Pedestrian trajectory · Motion planning · Long-short-term-memory network · A* search

1 Introduction

Nowadays, research about human-robot interaction is perpetually under the spotlight because of the emergence of complex artificial intelligence (AI). Researchers from all over the world have developed uncountable innovative inventions based on AI. For example, speaktoit assistants, such as Siri and Cortana, facilitate immediate information communication to the end-user; another instance is the introduction of intelligent robotics in industrial and manufacturing applications that improve efficiency and eliminate hazards. Autonomous driving belongs in this promising emergent domain studied by many researchers. Currently, autonomous vehicles remain plagued by limitations in performance, in particular, motion prediction, a necessary module to enable prophetic decision making. If we can develop a precise and effective prediction model, many new techniques will process significantly. For instance, autonomous vehicles need to get to the given destinations as soon as possible while avoiding potential hazards such as collisions with pedestrians and other vehicles. Without

© Springer Nature Singapore Pte Ltd. 2021
W. Cao et al. (Eds.): CONF-CDS 2021, CCIS 1513, pp. 105–129, 2021.
https://doi.org/10.1007/978-981-16-8885-0_10

predictions of the pedestrians' positions, the vehicle cannot decide how to behave next. Similarly, other service robots for household and industrial applications also need to be designed with the reduction of risk to people within its operational scope. Such prediction models are in severe demand in every situation where robots have to navigate themselves to goals in a human-occupied environment. By now, many studies have been conducted by researchers to address these problems, and while many models have been designed, this remains a significant challenge for researchers in autonomous guidance.

Beyond prediction models, motion planning is another crucial module for robots to make decisions. After predicting the positions of different kinds of obstacles, the vehicle has to decide on its acceleration and direction, which will further determine the motor power and the future positions of the vehicle. In this way, the car and pedestrians can interact actively and safely in the same environment.

In our study, we designed a long-short-term-memory (LSTM) [1] model for prediction, and we formulated the planning problem as a MDP [2] planning problem and designed an A* search algorithm for it. In addition, we connected these two models together and created a complete model to simulate a common situation where the car is asked to get the goal on a map with meandering pedestrians. To underscore the significance of the prediction, we compare the prediction results of our trained prediction model and a simple linear prediction which assumes the pedestrian will maintain the same velocity. In addition, we extended our comparisons to other existing models to find the advantages and drawbacks of our model on the public data set UCY [3]. (Fig. 1)

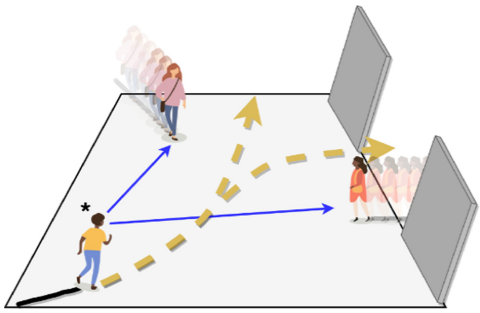

Fig. 1. Pedestrians trajectory prediction

2 Related Work

2.1 Prediction [4]

The software configurations of an intelligent vehicle can be generally summarized as three core components: perception, planning and control [5]. The perception module captures information presented in the environment such as positions, times, states, predictions, and sends it to other module in the system. The planning module employs the information gathered by perception to generate a full plan of actions toward the

goals. Such a plan is passed to the controller module to be conducted, which translates the plan into commands for the devices on the vehicle while employing different methods to reduce the error. Therefore, the vehicle is incapable of operation in absence of any of these three elements.

Traditional Models. Before the wide adoption of deep learning in this field, the researchers' general conception was to formulate this question based on physics or sociology and calculate the future positions of the pedestrians with some specific rules that were derived by them. For instance, an easy model is to assume that pedestrians will keep their current velocities or accelerations during the following time steps [6]. It becomes immediately apparent that these linear models are too simple to perceive complex interactions among pedestrians. Comparatively, recent models were developed with the ability to take such interactions into account. The vanguard of these approaches, the social-force model which was conducted by helbing, uses concepts from sociology to construct a system able to formulate the behaviors of human beings [7]. It models the interaction between humans by two virtual forces: an attractive force and a repulsive force. They bring such two forces into the prediction of future trajectories. This model has been proven to be effective and competitive in a number of scenarios and datasets [3, 8]. Based on this idea, some impressive and elaborate modifications have been proposed by other researchers [9, 10]. Moreover, by further observing into this task, researchers extracted more key information that affected pedestrians' trajectories from videos and took these factors into consideration in the model, and a variety of innovative formulations have been constructed by them. For instance, tay used a gussian process [11] to solve such a problem; antonini proposed a discrete choice model for prediction [12]; rodriguez came up with a way to analyze crowded scenes from videos; other models derived by lisotto [13] can perceive both interaction among pedestrians and between pedestrians and other crucial objects or scene scale; some other researches [14–16] analyzed people's behaviors via finding out the goal of pedestrians; some other combined pedestrians with similar behavior into clusters and analyzed them as a group [17, 18]. Though such formulated models are quick to train and run, the formulas invented by researchers lack flexibility since there are so many factors that can affect pedestrians' decisions, so the prediction will be terrible if the actual situation has not been formulated by the models. As a result, the neuron networks with abundant weights might have better performance.

Deep Learning Method. On the other hand, rather than using derived formulation to solve the problem, the mainstream has turned to teaching computers to generate a data-based comprehension of the pedestrian trajectory. However, the conventional fully connected neuron network might not be appropriate for spatial-temporal signals because they cannot establish a sense of order and remember the information extracted from previous time steps. Therefore, the recurrent neuron network (RNN) models were proposed to handle this problem. It shows its superiority, but improvement is necessary. So the variants of RNN, long short term memory (LSTM) [4] and gated recurrent units (GRU) [19], were invented as a response and proved their great performance. These modified RNN have been widely adopted in natural language processing (NLP) [20], speech recognition [21], trend forecasts, and machine translation [22]. As they did before, in this challenging field, LSTM and GRU have inspired many new

solutions with noble success. For instance, the sequence to sequence structure of LSTM has attained a relatively lower loss than many models [23]; the social LSTM [24], DAG-Net [25] and the social wagdat [26] model successfully set up connections among pedestrians to give the network chances to learn the relationship among agents; the ss-LSTM model [27] integrates the information from the scene and from the social reactions into the network; yagi [28] makes a prediction using first-person video data in order to take the ego-motions which can implicitly show intention of the people into consideration; shi, xiaodan's model [29] is also impressive because it accesses a way to deal with extremely crowded scenarios. These models all show different advantages when dealing with the public datasets [3, 8]. However, even though there are so many attempts to use deep learning in this task, sometimes the simplest constant velocity model can still outperform these complicated models [6], and it is much easier to train. Therefore, in the pedestrian's trajectory prediction problems, there are many different and competitive candidates and not a single "gold" approach exists that can handle all situations.

2.2 Planning [5]

The research of the motion planning module of robots is very popular with a long history. To meet the requirement for navigation in flexible real-life scenarios, the planning module is indispensable for both robots and automated vehicles because they both need to navigate in the flexible real-life scenes and complete their assigned tasks. The basic meaning of motion planning is to make decisions by the information that perception has captured, such as the positions of roads, other cars and pedestrians (what we study in this research). According to this framework, researchers have come up with a large amount of ideas and algorithms during the last few decades. These researches can be categorized into several groups; some of them are so classical models to the degree that a few examples were invented before the advent of computers, while some other algorithms are creative and impressive. Some groups of solutions are introduced below:

Graph Search Planning. The general idea of this branch is to find a series of future states in the state space for the vehicle by searches or other algorithms. When the robot executes along these planned states, it will get optimal reward. A classic example of this kind of algorithm is shown in Fig. 2, what the graph search algorithm does is to generate a search tree that consists of nodes of states based on such a graph; then the nodes in the trees are checked and evaluated until the goal, or final nodes, are met. The pioneer of these algorithms is the dijkstra algorithm [30–32], it can find a single-source shortest path. Inspired by this fundamental algorithm, many other algorithms emerged. DFS (Depth first search) [33] and BFS (Breadth first search) [34] are two simple instances. Then UCS (Uniform cost search) [35], which does BFS according to a cost function in order to reduce the nodes that need to be expanded and is able to make sure that the result is optimal, is implemented. The differences between UCS and BFS are shown in Fig. 3 and Fig. 4. The blue dots are the nodes in the search graph, and the red dots are the goal states. Moreover, extended from Dijkstra algorithm, A* [36] was invented. This search scheme is quicker than other existing methods because of the

adoption of heuristics: this algorithm uses the sum of the past costs and the estimation of the distance to the goal as a criterion to find and evaluate the optimal path. Therefore, designing the cost function and heuristics is the most important and challenging task to perform such an algorithm. Across a long term of researching, differently designed variants of A* such as the anytime D* [37], the dynamic A* [36], the Theta* [38], and the field D* [39] were employed in automated robots and vehicles. However, though the graph search algorithms can find the best path with limited information, they also have obvious drawbacks: they plan the path in discrete nodes instead of a smooth track. In addition, they are also time-consuming when dealing with intricate situations because they need to expand and enumerate many nodes in the searching tree, while computational time is extremely crucial in automated driving that has to take reaction to the changes in the environment and replan a new path in a moment. In our work, we design a cost function and heuristics for the vehicle-pedestrian scene.

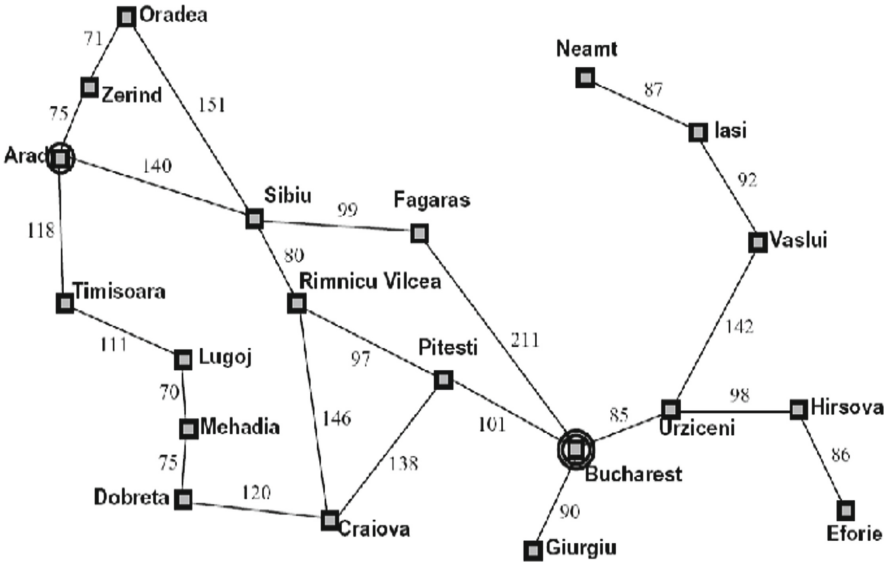

Fig. 2. Traveling in Romania—a classical example of Graph Search

Interpolating Curve Planning. The two approaches discussed above are generally recognized as global planners that can only generate approximate and discrete paths. To complete the solution, interpolating curve planning is applied. Interpolating means to insert data into the rough data given by global planners and output a smooth and continuous trajectory. The Computer-aided Geometric Design (CAGD) [40] is a fundamental technique in the application of this task. Researchers use different curves like clothoid curve [41, 42] and polynomial curve [43, 44] to fit the points and make the path smoother.

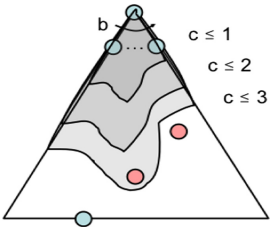

Fig. 3. The UCS search

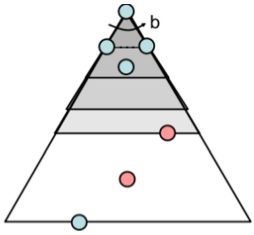

Fig. 4. The BFS search

Interpolating Curve Planning. The two approaches discussed above are generally recognized as global planners that can only generate approximate and discrete paths. To complete the solution, interpolating curve planning is applied. Interpolating means to insert data into the rough data given by global planners and output a smooth and continuous trajectory. The Computer-aided Geometric Design (CAGD) [40] is a fundamental technique in the application of this task. Researchers use different curves like clothoid curve [41, 42] and polynomial curve [43, 44] to fit the points and make the path smoother.

3 Problem Statement

3.1 Overall Procedure

As Fig. 5 shows, to address the task of navigating, a robot has to be able to complete this looping procedure. At first, the camera that is deployed at the front of the robot captures information from the environment in the form of video. Then, through a Convolutional Neuron Network (CNN) [45], the robot perceives features of the environment such as the historical positions of pedestrians and other objects. This information is sent to the prediction model to predict future positions which are crucial to the planning algorithm. After the planning module has solved an optimal path for the robot, the controller is called to generate a sequence of power commands to the motors to move the robot by only one node of the path, and the rest of the path is abandoned. This is the principle of Model Predictive Control (MPC). This trick is conducted to ensure the planned path depends on true features since the features in the environment undergo constant change and more recent paths can generate more accurate predictions. Finally, when the robot has achieved the next state, factors in the environment, including pedestrians' positions and the robot itself, will change, so a recapture of the information is necessary. As this loop is running, the robot will gradually approach its goal until it achieves it. Our system is represented with the green block in Fig. 5, and it can convert the extracted positions of pedestrians into a proper path for the vehicle given the goal and other stable barriers in the environment.

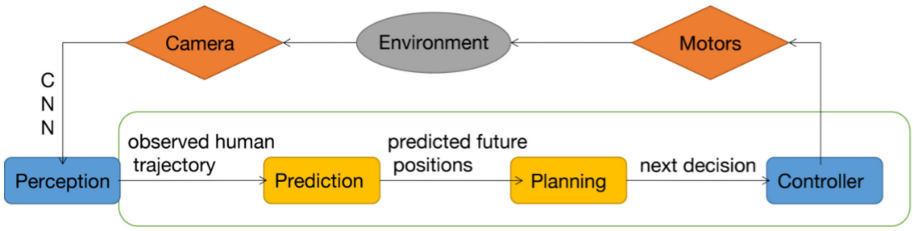

Fig. 5. The overall procedure of the system

3.2 Prediction

In the prediction module, a sequence of the historical positions is translated and read by a computer, and another sequence of predicted future positions returns. Therefore, what our prediction model does can be summarized with the following formula:

$$(\widehat{x},\widehat{y})_{t+1},(\widehat{x},\widehat{y})_{t+1}\cdots(\widehat{x},\widehat{y})_{t+T}=f\big((x,y)_{t-H+1},(x,y)_{t-H+2},\cdots(x,y)_{t}\big) \quad (1)$$

Where the f (x) stands for the whole function of prediction; $(x,y)_t$ represents the predicted value of the x and y position at the time t; T is the time horizon the models will predict and H is the length of the historical sequence that is put into the network. Therefore, the prediction problem is a sequence-to-sequence prediction with input and output features of size 2 (x and y position).

3.3 Planning

We expect our robot to do the things shown in Fig. 6: to generate a sequence of actions that result in a path to the goal among different kinds of barriers. The orange dots are the prediction of pedestrians, while the orange lines represent the true paths. The blue line is the planned path of the vehicle, and the blue dots are the actual positions at each time step.

In the planning algorithm, since the next decision only depends on the current state of the agent and the predicted information, this problem can be considered as a Markov Decision Process (MDP) [5]. Thus, the state space and action space have to be defined. In this specific question, the state is actually the attributes of the robot in a certain situation, and our definition is that every state is a vector of size 5: t (current time step), x (x position), y (y position), v (the current velocity of the agent), d (the direction that the robot faces, measured in radians). As shown in the equation:

$$S=(t,x,y,v,d) \quad (2)$$

On the other hand, the action space is the range of choices available to the robot given the current state to transform to the next state. By simplifying the robot as a point and considering about real scene when we drive, we know that the choice of path at a specific state is actually the choice of acceleration and turning angle. By taking these actions, the robot can reach any state in the state space. So, the action space is defined as follows.

$$a_s = (a, \theta) \tag{3}$$

Moreover, via the prediction module, the robot has received the predicted future trajectories of the pedestrians, we represent it with $(Xp, Yp)_{t+1:t+T}$, which means the estimation of pedestrians' positions during the following T (predicted step) steps. Therefore, we can design a cost function to evaluate whether a single action is a good choice. In this way, the algorithm will be able to generate the planned path step by step because we can update the current state with the chosen action without interference from the previous decisions, and the entire process can be represented by this formula.

$$(a_{t:t+T}, \delta_{t:t+T}) = argmin\left(C\left(S_t, S_{goal}, \widehat{(X_p, Y_p)}_{t+1:t+T}\right)\right) \tag{4}$$

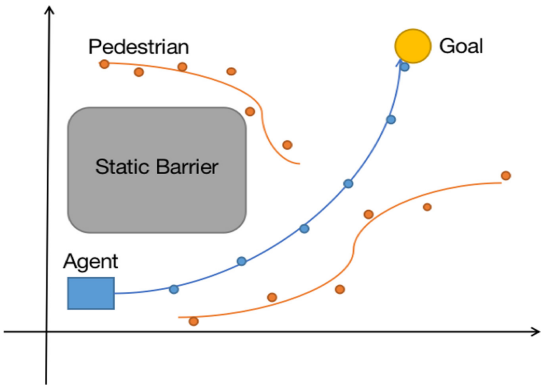

Fig. 6. The situation of our planning program

Where $a_{t:t+T}$ and $\delta_{t:t+T}$ is the planned accelerations and diversions (the action) of the agent in the following T time steps. T is the steps we want to plan (equal to our predicted output); C() is the cost function and St stands for the state of the agent at the time t (current time).

This formula means that the sequence of accelerations and diversions from the source toward the goal in a certain time period is determined by minimizing a cost function that judges and weighs different factors including the risk of colliding with pedestrians. More details regarding the design of the cost functions and the planning algorithm are given in the following sections.

4 Method

4.1 Heuristic and Cost Functions

The heuristic is just a simple estimation of how much work the agent need to do to achieve the goal. However, we should bear in mind a critical rule when designing a heuristic: the heuristics must be admissible and consistent if we want to guarantee the optimality of the path. Admissible and consistent: $\forall(H_i - H_j) \leq C_{ij}$.

Where H means the heuristic of node i and Cj represents the actual cost form node i to node j. As a result, the heuristic will never overestimate the actual distance and the A-score will never decrease along the expanded nodes. In this specific question, our heuristic is defining as:

$$H = \begin{cases} \infty(0 \times ffff) & (S \in ICS) \\ W_H \times \left(\left(x - x_{goal}\right)^2 + \left(y - y_{goal}\right)^2 \right) & (S \notin ICS) \end{cases} \tag{5}$$

Where W_H means a constant weight. We used the square of Euclidean Distance (the straight line distance) to the goal to estimate the cost because this measurement never overestimates the distance, so our algorithm can be optimal. Moreover, we also adopt a trick on H value to reduce the nodes that need to be expanded. We used a concept called Inevitable Collision State(ICS) [46, 47] the state that no matter what the vehicle decides to do, collision, or at least extremely close contact, is inevitable according to the current prediction. Therefore, if the current state is in ICS, we can feel free to set its H-value to a value that is close to infinity because will definitely not choose this path. This can prevent the search from further expanding unnecessary nodes.

Besides H value, G value is also crucial to finding the best path. Since it is the measurement of the total cost from origin to current states, we summarized several costs that the agent will face below and design our G function based on these rules:

- The speed of the vehicle should not be too much bigger than the speed limitation, because over-speeding is always illegal and can be a potential safety threat.
- The vehicles should accelerate or decelerate as minimally as possible, frequent changes of velocity will make passengers uncomfortable.
- Too frequent turnings are also unfriendly to passengers and may giddy them; in addition, more turnings can make the demand of the accuracy of the controller and the motors stricter.
- The vehicles should maintain distance from pedestrians as close contact is a potential risk for collision and the prediction model cannot be relied upon completely.
- In any situation, saving time while reducing potential hazards is the preferred outcome, so the planned path should be as short as possible to a time scale.

We designed G value as the sum of several functions which represent these costs with their weights. Concrete formulas are listed below accordingly (all W is the weight):

$$C_{vlim} = \begin{cases} 0 & v \le v_{lim} \\ W_{vlim} \times (v - v_{lim})^2 & v > v_{lim} \end{cases} \tag{6}$$

Where v_{lim} stands for the speed limitation. This equation is easy to understand, if the car does not overspeed, the cost should be zero, and the car does have excessive velocity, the cost will increase at a square rate.

$$C_{acc} = W_{quareacc} \times a^2 \tag{7}$$

Where a is the acceleration that need to be executed to the state that is being evaluate.

$$C_{turn} = W_{turn} \times \theta^2 \tag{8}$$

Where 6 stands for the turning angle that need to be conducted to the next state.

$$C_{coll} = \sum_{i=1}^{i \le p} W_{coll} \times e^{-k*\left((x-x_i)^2 + (y-y_i)^2\right)} \tag{9}$$

Where k is another parameter that can make the curve less steep, and p is the number of pedestrians that the agent is able to detect; x_i and y_i are the predicted positions of pedestrians i at current time step. We use exponential function to describe this cost because this function increases at a high rate as x is close to zero, which can match our situation that extremely close distance is absolutely Forbidden.

$$C_{time} = W_{time} \times (\Delta t)^m \tag{10}$$

Where Δt is the time difference between the current state and the initial state. m is also a parameter and m This function increases rapidly at the beginning and then becomes gentle. This is because we want the agent to feel the urgency of the time at the beginning of the search, or it will choose to remain still as time lost is otherwise irrelevant to its decision-making.

By this definition and a time-consuming adjustment of weight, we have built a metric which helps our algorithm to search effectively. (Fig. 7)

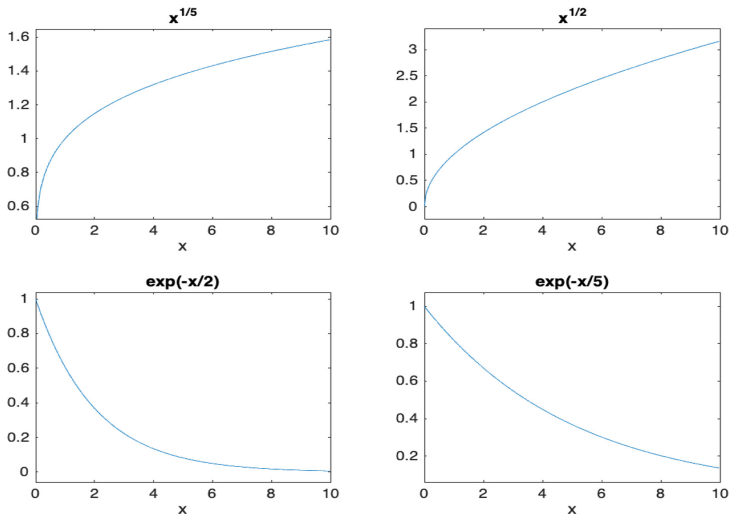

Fig. 7. The graphs of some example functions

4.2 Transitional Function

The algorithm requires a function to update the attributes of the current state after deciding the acceleration and diversion. In our situation, t, v, and d is easy to derive:

$$t_{i+1} = t_i + dt \tag{11}$$

$$v_{i+1} = v_i + a \tag{12}$$

$$d_{i+1} = d_i + \theta \tag{13}$$

Where d_t is the time of a time step, and we define a and θ as using the unit time steps. In addition, to make our model simpler and reduce the computational time, we assumed the car to be a mass point in our search, and the time that the car needs to turn its steering angle is ignored. Since then, the position of the next state after choosing the action has also become uncomplicated:

$$x_{t+1} = x_t + v_i * dt + \left(\frac{1}{2} a dt^2 * \cos(d + \theta)\right) \tag{14}$$

$$y_{t+1} = y_t + v_i * dt + \left(\frac{1}{2} a dt^2 * \sin(d + \theta)\right) \tag{15}$$

In this way, the transition between states is available.

4.3 Search Procedure

In our algorithm, we used Priority Queue [48] to store the fringe nodes and easily select the nodes with the lowest A-score. In addition, we also applied a dictionary to store the parents of the nodes and at the same time serve as a close list that store expanded nodes which should not be expanded again. The procedure of our A* algorithm is shown below, and it is also shown in Fig. 8.

Step 1: Put the origin into the Priority Queue with its A-score as priority.

Step 2: Take the state with the least priority out of the queue.

Step 3: If this node is already being expanded, skip and start from step 2.

Step 4: If the chosen state is a goal state or there is no prediction for the algorithm to continue (have search designed steps), save this node as skip to step 7.

Step 5: For every choice that the agent can take, generate the resulting state and its A-score, push all of them into the priority queue.

Step 6: Set the current state as the parent of all the generated states, return to step 2.

Step 7: Use the saved node to find back to the origin and get a path. update the origin to the first state of the path. If the origin is the goal, then our algorithm is done, but if not, we should re-predict pedestrians' trajectory, clear the queue and parent dictionary, and return to step 1.

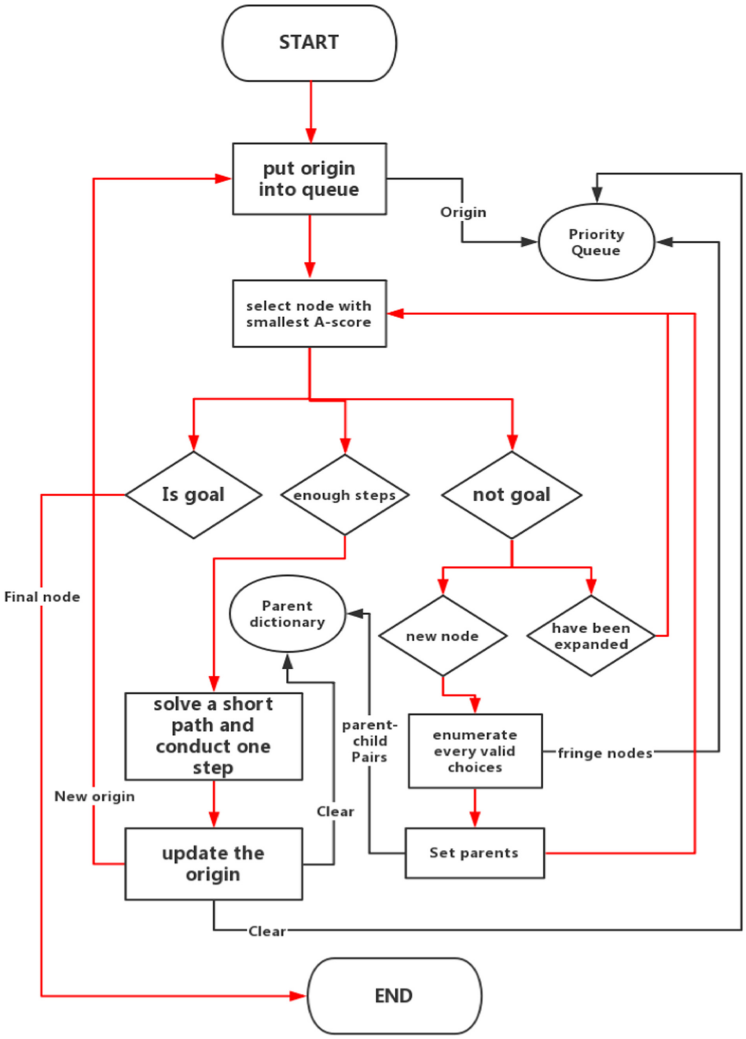

Fig. 8. Procedure of A*

Dataset. We use datasets that originate from two sources. First, we collected some videos of people's footage by drones. After that, we adopted CNN (convoluted neuron network) [45] to translate the videos into historical sequences of positional information. This dataset has a time step of 0.1 s. Moreover, we also utilized the data that has been published by the university of cyprus [3]. This dataset includes many sequences of pedestrians' trajectory, which is represented by positions. There is also a feature called the gaze direction, standing for the direction that pedestrians look. However, in order to maintain the same number of features with the previous dataset, we have abandoned this feature.

Training Configurations. In our training, we modified our hyper parameters for a long time in order to optimize the performance of our models. At last, we found a set of balanced hyper parameters which is listed below:

- Learing rate: 0.01
- Batch size: 25
- Train test validation data ratio: 7: 2: 1
- Epoch: 100
- Activation function: ReLU
- Loss function: MAE (Mean Square Error)
- Optimizer: Adam

With these hyper parameters, our models are able to achieve a relatively good performance and at the same time keep the computational time acceptable.

Prediction Performance. We trained and tested our model with the dataset collected by ourselves. We pre-processed the data and inputted it to our models. The visualizations of the prediction are shown below. The red lines are the historical path; the blue ones are the ground truth. Different colors of scatters represent the prediction by models. Yellow is the simple constant velocity model's, purple stands for our multi-step prediction, while green means our feedback model (autoregressive model). These selected graphs are the typical cases that can represent the performance of our models.

In cases (1) and (2), all predictions from all models are precise to an acceptable standard. This is because the behaviors of the pedestrians in these cases are simple: they almost walk along a straight line. Therefore, the constant velocity model can also have acceptable performance (Figs. 9, 10, 11 and 12).

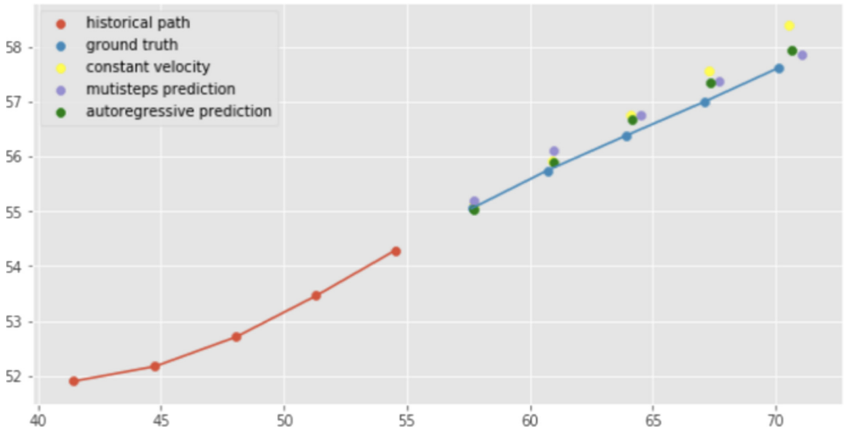

Fig. 9. Case (1)

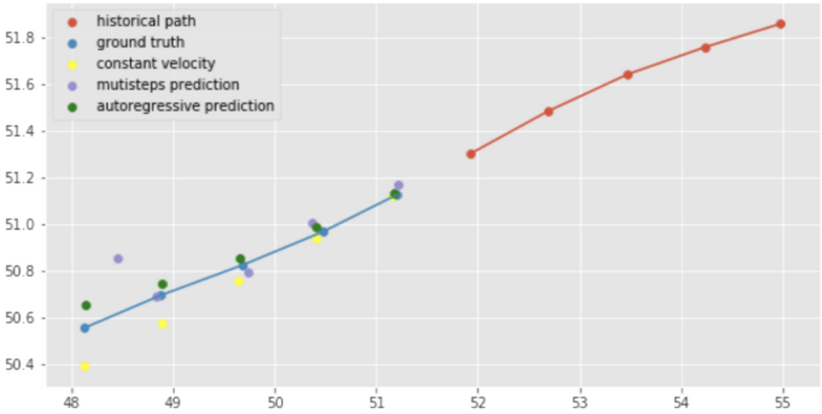

Fig. 10. Case (2)

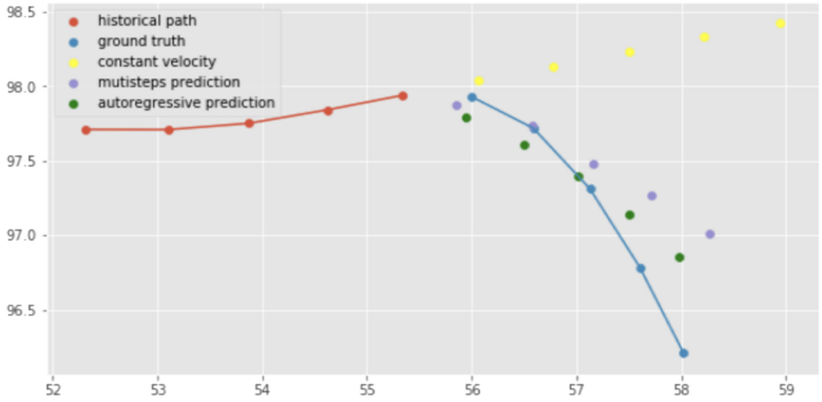

Fig. 11. Case (3)

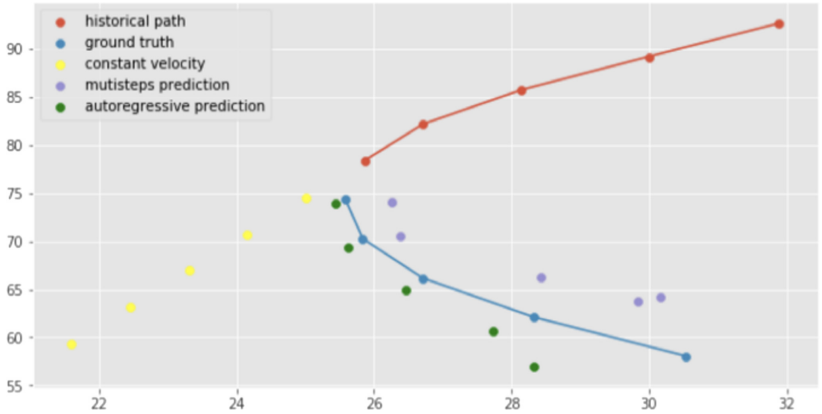

Fig. 12. Case (4)

However, case (3) and (4) show that the superiority of our prediction models is apparent. These two pedestrians suddenly turn right or turn back, and not unsurprisingly, the simple models have no ability to capture their tendencies to turn. However, both of our two models achieve that and generate an approximate prediction.

The judgment on the change of directions from the pedestrians is a far more crucial part than predicting the exact positions of them in autonomous driving since the car always deals with questions like whether the pedestrians will pass the intended path of the car instead of how far they will go. In addition, as pedestrians' speeds are significantly lower than those of cars, the value of error does not matter a lot when we try to decide which model we should use in our search model. As shown in cases (5), (6), (7) and (8), though the pedestrians occasionally behave in unexpected ways, our single-step model prediction successfully predicted the future direction while the other two methods get lost. Moreover, in case (9), the single-step model is also more accurate than the multi-steps model; this superiority indeed exists in the majority of our data-set. As a result, we decided to use the single-step model in our planning (Figs. 13, 14, 15, 16 and 17).

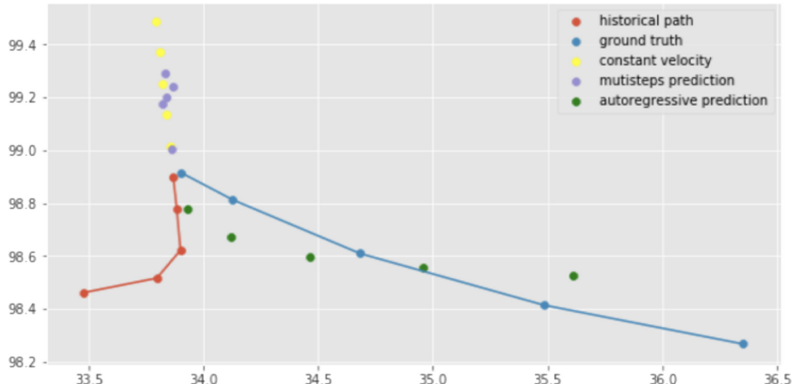

Fig. 13. Case (5)

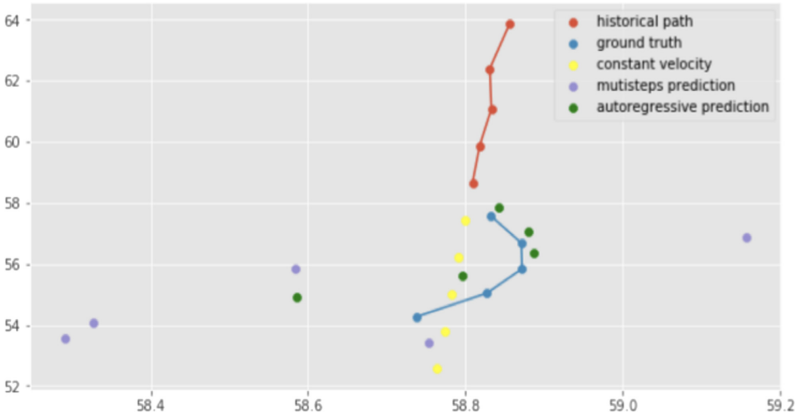

Fig. 14. Case (6)

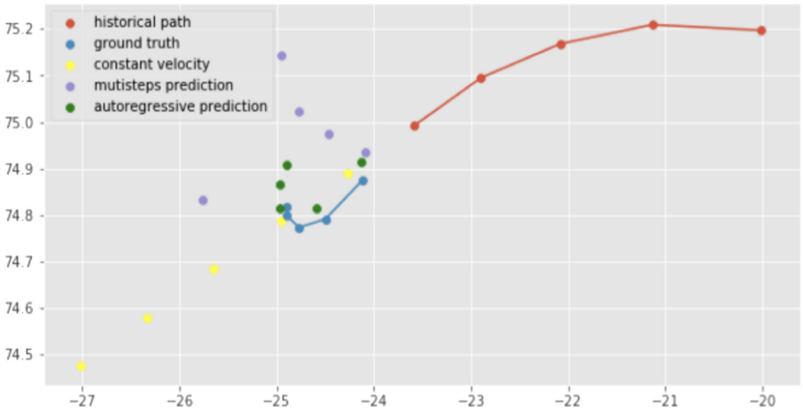

Fig. 15. Case (7)

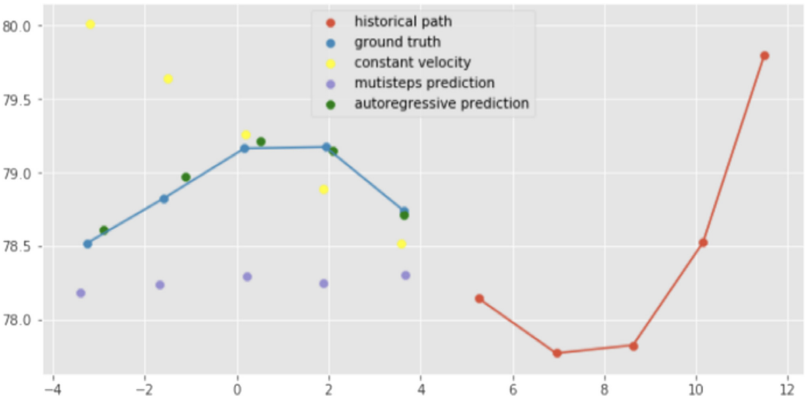

Fig. 16. Case (8)

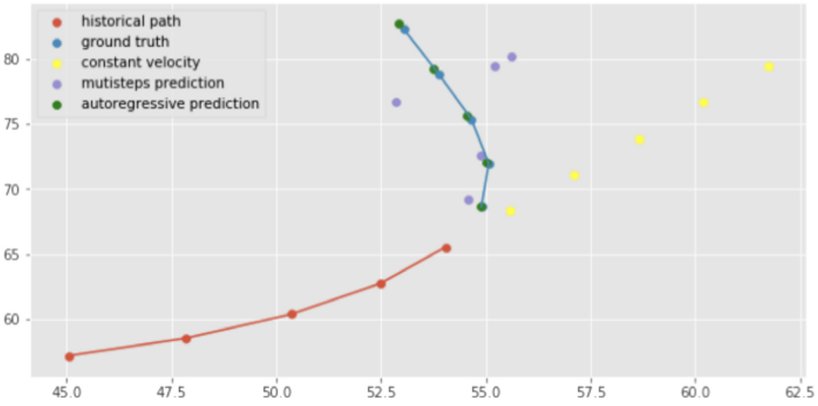

Fig. 17. Case (9)

However, our model still has many limitations. In cases (10) and (11), the pedestrians' behaviors are erratic. The first one takes a sharp turn and accelerates quickly; the second one waits and abruptly turns around. As a result, all of our attempts failed to comprehend such behaviors. This revealed one of our models' limitations that we can only predict a person's path with only his or her historical paths. Hence, it is impossible for our models to successfully take the reactions that he or she has because of other people or objects into account (Figs. 18 and 19).

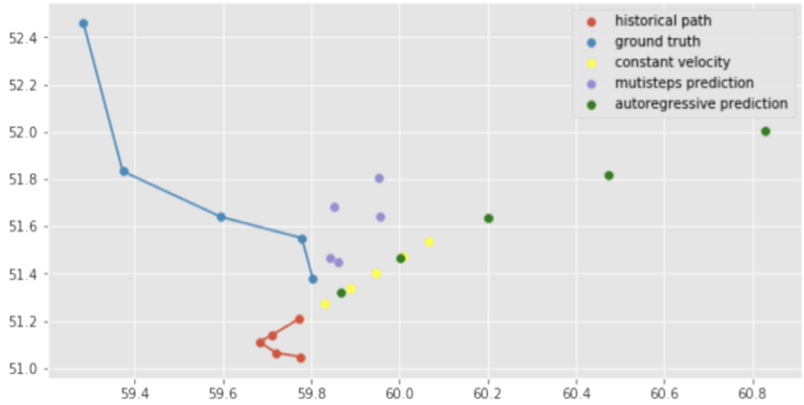

Fig. 18. Case (10)

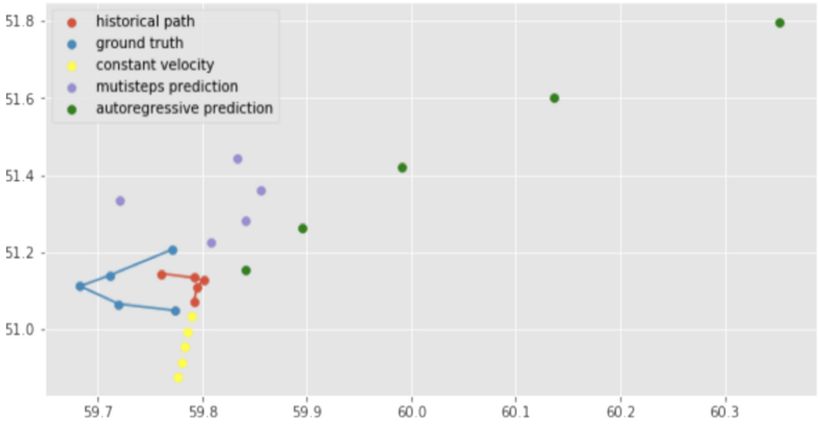

Fig. 19. Case (11)

Comparison Among Our Prediction Models and Other Models. In addition to testing the performance of our prediction model, we also evaluated the MAE (mean absolute error) and compared it to other models that list below.

Table 1. Data comparison among models

error (m) models / data	Line.	SF.	Sin.	S-LSTM
Zara_1[3]	0.47	0.4	0.26	0.22
Zara_2[3]	0.45	0.4	0.25	0.25
UCY[3]	0.57	0.48	0.46	0.27
Total	1.49	1.28	0.97	0.74

- Linear model [6] (Line.): the simple model that assumes the velocities of pedestrians are constant.
- Social-Force model [9] (SF.): the model that formulates the interaction among people.
- Our Single-step model (Sin.): Since the Single-step model outperforms the multi-steps models, we use it to compare with other models.
- Social-LSTM [24] (S-LSTM.): the LSTM network that is specially designed to take the relationship among pedestrians into consideration.

The loss is shown in the following table. (Zara_1, Zara_2, and UCY datasets are all form University of Cyprus, as mentioned in the dataset subsection) (Table 1).

Unsurprisingly, the Linear model has the greatest error generation among these models; the Social-Force model is better, but it is still too inaccurate to be put into practice. Our LSTM, which uses the traditional structure, was a little worse than the Social-LSTM [24]. However, since our model has fewer parameters, it is an advantage that we can complete our training faster. Moreover, our model resulted in undesirable outcomes on the UCY data set while the Social- LSTM performed far above expectations. We think the reason is that the UCY dataset is much more crowded than other data. Therefore, our model loses its competitiveness because of its disability to learn the interactions.

Simulations. We chose some pedestrians from our data set and ran a simulation of our planning algorithm. In our situation, we used several pedestrians and static barriers to block the car as it planned its path to the goal which had been set. The following figures show the paths of cars and pedestrians in some cases, and we used the same color map in order to make a comparison between different paths easier. The red dot represents the goal, and the block in the red rectangle represents the static barrier. The 3D plot also displays the paths of pedestrians and the car, and the z-axis is time (higher means longer from the initial time).

On track (1), the car has to the pedestrians because of the existence of the static barrier. After turning, it accelerates toward the goal. As shown in Fig. 21, the speed of the car goes up without deceleration. In addition, in Fig. 22, the distances between the car and the pedestrians never become smaller than our predefined safety threshold (lower than 3 m) to cause the risk of collisions. As a result, though the path looks reckless, the car actually prioritized safety above time (Figs. 20 and 23).

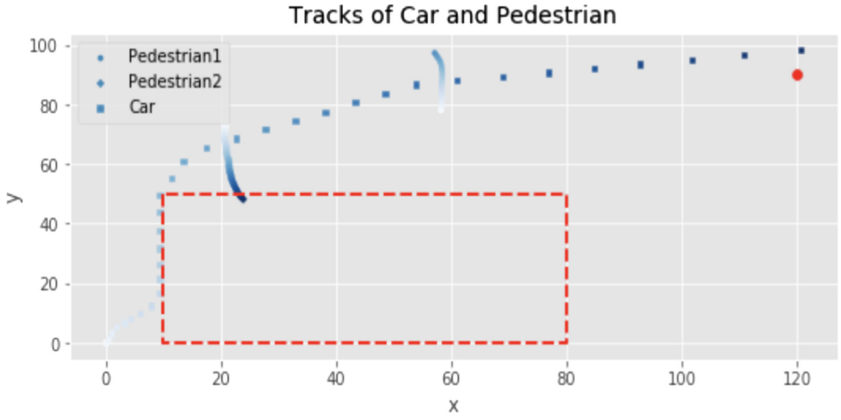

Fig. 20. Track (1)

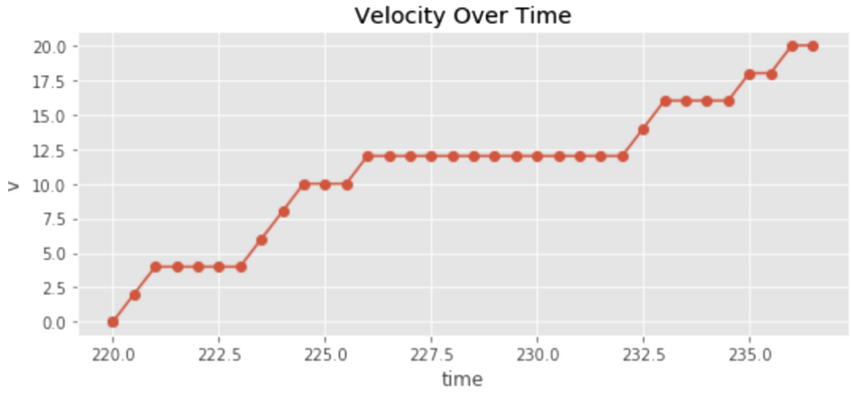

Fig. 21. Track (1) v-t graph

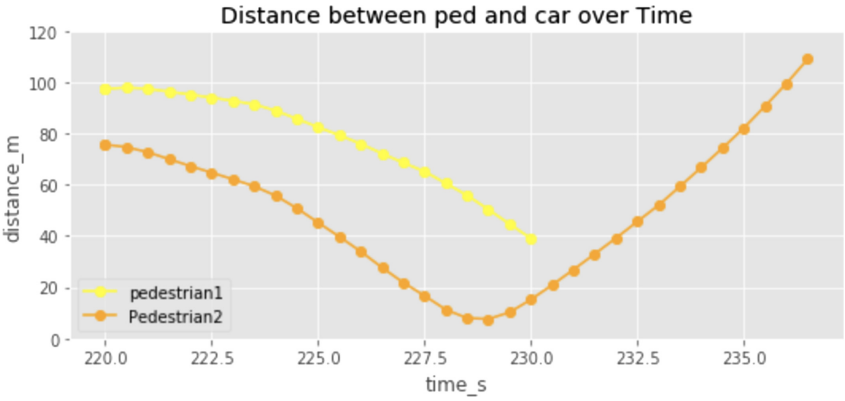

Fig. 22. Track (1) distance-t graph

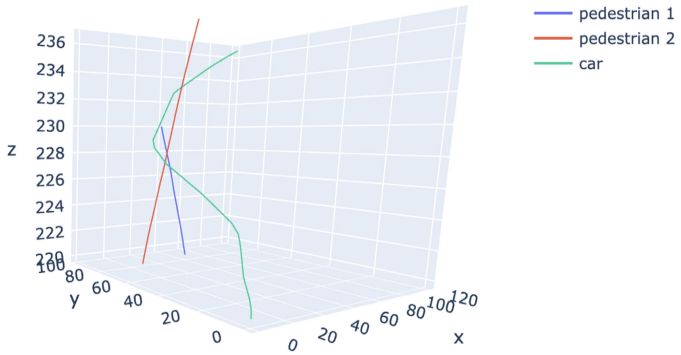

Fig. 23. Track (1) 3D graph

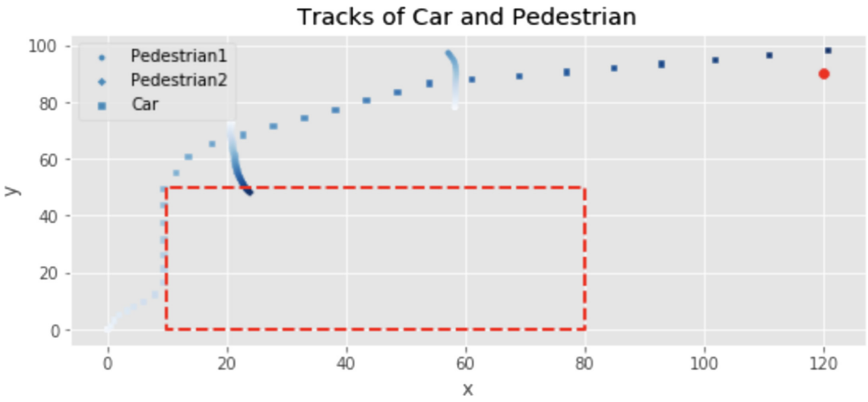

Fig. 24. Track (2)

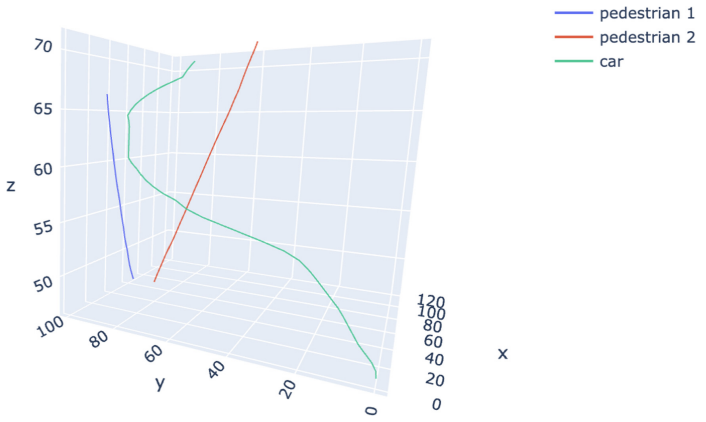

Fig. 25. Track (2) 3D graph

On the other hand, on track (2), as shown in Fig. 24, instead of driving straight to the goal, it chooses to stop and wait for the pedestrians in front of it to clear out. This decision reflects that the car had perceived the risk of collision. Therefore, our algorithm has the ability to answer the simple but crucial question—of who should go first, the car or the pedestrians? (Fig. 25).

Finally, to compare the algorithm's performance when different predictions are used to generate a path, we designed a metric function to evaluate how appropriate the paths are. This function consists of measurements of the paths from different dimensions, including the total distance the car travels, the risk of collisions between pedestrians and the car at every time step, etc. These functions are all added up with their weights. The expression of the metric is shown below:

$$\mathrm{M} = W_{time} * t + W_{dist} * s + W_{acc} * \sum a_i^2 + W_{turn} * \sum d_i^2 + W_{coll} * \sum \sigma_i$$

Where different W stand for different weights and are different from those used for planning, t is the total time cost, s is the total distance. a_i and di are the accelerations and the turning angle at time $step_i$. a is the function to evaluate the risk of collision, defined by the following expression:

It is actually almost the same as the C_{coll} in the A* search. The only difference is that x_i and y_i stand for the true positions of pedestrians instead of predicted ones. Therefore, the metric can evaluate the true risk of the path (Table 2)

Apparently, the path generated by the LSTM prediction outperforms the others because in all of the cases tested, the cost metric of the paths with LSTM prediction is lower than those with simple prediction algorithms. Therefore, we can draw the conclusion that a more accurate prediction can indeed assist the planning algorithm in generating a better path.

Table 2. The metrics of planned paths with simple linear prediction and our LSTM prediction in some cases

metric \ prediction / number	simple prediction	LSTM prediction
case1	234312.45	206205.87
case2	99528.83	81584.68
case3	56049.12	54463.79
case4	46537.16	43987.16
case5	74056.99	73593.43
Total	510484.55	459834.93

5 Conclusions

When an autonomous car meets a pedestrian at a crossing, should the car go first or the pedestrian? This problem has to be solved to approach our self-driving dream. Focusing on this problem, we use two LSTM models and A* to designed two modules that are indispensable to autonomous driving and robotic navigation: prediction and planning. Although these two parts have received significant coverage previously, by research, we effectively built a connection between these two sections on our algorithms. These experiments show that computers can generate optimal paths with positive outcomes. It also illustrates that these algorithms have a promising future. Admittedly, our methods are relatively easier than those recent researches have proposed. We did not take the interactions among pedestrians into account like what the mainstream of researches tries to do; we did not use the interpolation curve to make our path smooth rather than discrete. Nevertheless, we have proven that using a better scheme for prediction in motion planning matters a lot by combining and applying previous researchers' intelligence.

References

1. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
2. Koenig, S., Simmons, R., et al.: Xavier: a robot navigation architecture based on partially observable markov decision process models. Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems, no. partially, pp. 91–122 (1998)
3. Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example, in Computer graphics forum, vol. 26, pp. 655–664, Wiley Online Library (2007)
4. Rudenko, A., Palmieri, L., Herman, M., Kitani, K.M., Gavrila, D.M., Arras, K.O.: Human motion trajectory prediction: a survey. Int. J. Robot. Res. **39**(8), 895–935 (2020)
5. González, D., Pérez, J., Milanés, V., Nashashibi, F.: A review of motion planning techniques for automated vehicles. IEEE Trans. Intell. Transp. Syst. **17**(4), 1135–1145 (2015)

6. Schöller, C., Aravantinos, V., Lay, F., Knoll, A.: What the constant velocity model can teach us about pedestrian motion prediction. IEEE Robot. Automation Letters **5**(2), 1696–1703 (2020)
7. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**(5), 4282 (1995)
8. Pellegrini, S., Ess, A., Schindler, K., Van Gool, L.: You'll never walk alone: modeling social behavior for multi-target tracking. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 261–268, IEEE (2009)
9. Yamaguchi, K., Berg, A.C., Ortiz, L.E., Berg, T.L.: Who are you with and where are you going? In: CVPR 2011, pp. 1345–1352, IEEE (2011)
10. Pellegrini, S., Ess, A., Van Gool, L.: Improving data association by joint modeling of pedestrian trajectories and groupings. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) Computer Vision–ECCV 2010. ECCV 2010. Lecture Notes in Computer Science, vol. 6311. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15549-9_33
11. Tay, M.K.C., Laugier, C.: Modelling smooth paths using gaussian processes. In: Laugier, C., Siegwart, R. (eds.) Field and Service Robotics. Springer Tracts in Advanced Robotics, vol. 42. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-75404-6_36
12. Antonini, G., Bierlaire, M., Weber, M.: Discrete choice models of pedestrian walking behavior. Transp. Res. Part B: Methodol. **40**(8), 667–687 (2006)
13. Lisotto, M., Coscia, P., Ballan, L.: Social and scene-aware trajectory prediction in crowded spaces. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, p. 0 (2019)
14. Gong, H., Sim, J., Likhachev, M., Shi, J.: Multi-hypothesis motion planning for visual object tracking. In: 2011 International Conference on Computer Vision, pp. 619–626, IEEE (2011)
15. Huang, C., Wu, B., Nevatia, R.: Robust object tracking by hierarchical association of detection responses. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) Computer Vision – ECCV 2008. ECCV 2008. Lecture Notes in Computer Science, vol. 5303. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88688-4_58
16. Kretzschmar, H., Kuderer, M., Burgard, W.: Learning to predict trajectories of cooperatively navigating agents. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 4015–4020 (2014)
17. Chandra, R., et al.: Forecasting trajectory and behavior of road-agents using spectral clustering in graphlstms. IEEE Robot. Automation Lett. **5**(3), 4882–4890 (2020)
18. Leal-Taixé, L., Pons-Moll, G., Rosenhahn, B.: Everybody needs somebody: modeling social and grouping behavior on a linear programming multiple people tracker. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV workshops), pp. 120–127, IEEE (2011)
19. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014). arXiv preprint arXiv:1412.3555
20. Manning, C., Schutze, H.: Foundations of statistical natural language processing. MIT Press (1999)
21. Graves, A., Mohamed, A.-R., Hinton, G.: Speech recognition with deep recurrent neural networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 6645–6649, IEEE (2013)
22. Koehn, P.: Statistical machine translation. Cambridge University Press (2009)
23. Park, S.H., Kim, B., Kang, C.M., Chung, C.C., Choi, J.W.: Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1672–1678, IEEE (2018)

24. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: human trajectory prediction in crowded spaces. In: Proceedings of the IEEE conference oncomputer vision and pattern recognition, pp. 961–971 (2016)
25. Monti, A., Bertugli, A., Calderara, S., Cucchiara, R.: Dagnet: double attentive graph neural network for trajectory forecasting (2020). arXiv preprint arXiv:2005.12661
26. Li, J., Ma, H., Zhang, Z., Tomizuka, M.: Social-wagdat: interaction-aware trajectory prediction via wasserstein graph double-attention network (2020). arXiv preprint arXiv:2002.06241
27. Xue, H., Huynh, D.Q., Reynolds, M., Ss-lstm: a hierarchical lstm model for pedestrian trajectory prediction. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1186–1194. IEEE (2018)
28. Yagi, T., Mangalam, K., Yonetani, R., Sato, Y.: Future person localization in first-person videos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7593–7602 (2018)
29. Shi, X., Shao, X., Guo, Z., Wu, G., Zhang, H., Shibasaki, R.: Pedestrian trajectory prediction in extremely crowded scenarios. Sensors **19**(5), 1223 (2019)
30. LaValle, S.M.: Planning algorithms. Cambridge University Press (2006)
31. Hwang, J.Y., Kim, J.S., Lim, S.S., Park, K.H.: A fast path planning by path graph optimization. IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans **33**(1), 121–129 (2003)
32. Li, Q., Zeng, Z., Yang, B., Zhang, T.: Hierarchical route planning based on taxi gps-trajectories. In: 2009 17th International Conference on Geoinformatics, pp. 1–5, IEEE (2009)
33. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
34. Beamer, S., Asanovic, K., Patterson, D.: Directionoptimizing breadth-first search. In: SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, pp. 1–10, IEEE (2012)
35. Felner, A.: Position paper: Dijkstra's algorithm vs. uniform cost search or a case against Dijkstra's algorithm (2011)
36. Stentz, A.: Optimal and efficient path planning for partially known environments. In: Hebert, M.H., Thorpe, C., Stentz, A. (eds.) Intelligent Unmanned Ground Vehicles. The Springer International Series in Engineering and Computer Science (Robotics: Vision, Manipulation and Sensors), vol. 388. Springer, MA (1997). https://doi.org/10.1007/978-1-4615-6325-9_11
37. Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., Thrun, S.: Anytime search in dynamic graphs. Artif. Intell. **172**(14), 1613–1643 (2008)
38. Nash, A., Daniel, K., Koenig, S., Felner, A.: Theta^*: Anyangle path planning on grids. AAAI **7**, 1177–1183 (2007)
39. Ferguson, D., Stentz, A.: Using interpolation to improve path planning: the field d* algorithm. J. Field Robot. **23**(2), 79–101 (2006)
40. Farin, G.: Curves and surfaces for computer-aided geometric design: a practical guide. Elsevier (2014)
41. Brezak, M., Petrovic, I.: Real-time approximation of clothoids with bounded error for path planning applications. IEEE Trans. Rob. **30**(2), 507–515 (2013)
42. Fraichard, T., Scheuer, A.: From reeds and shepp's to continuous-curvature paths. IEEE Trans. Rob. **20**(6), 1025–1035 (2004)
43. Lee, J.-W., Litkouhi, B.: A unified framework of the automated lane centering/changing control for motion smoothness adaptation. In: 2012 15th International IEEE Conference on Intelligent Transportation Systems, pp. 282–287, IEEE (2012)

44. Petrov, P., Nashashibi, F.: Modeling and nonlinear adaptive control for autonomous vehicle overtaking. IEEE Trans. Intell. Transp. Syst. **15**(4), 1643–1656 (2014)
45. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: a convolutional neural-network approach. IEEE Trans. Neural Netw. **8**(1), 98–113 (1997)
46. Fraichard, T., Asama, H.: Inevitable collision states - a step towards safer robots? Adv. Robot. **18**(10), 1001–1024 (2004)
47. Martinez-Gomez, L., Fraichard, T.: An efficient and generic 2d inevitable collision state-checker. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 234–241 (2008)
48. Slater, G.S.C., Birney, E.: Automated generation of heuristics for biological sequence comparison. BMC Bioinf. **6**(1), 31 (2005)