Path planning for a mobile robot in an unknown environment using D* lite: Homemate robot

Jakub Tomasek

CONTENTS

I. INTRODUCTION

A. Homemate robot

1) Korus project: Homemate is a mobile manipulation robot developed as part of KORUS (Korea + US) project mainly by Yujin and Intelligent Systems Research Institute (ISRC) at Sungkyunkwan university; see [1] for more information about the project.

Figure I.1. Homemate robot

Currently, already a third generation of the robot is available.

The robot belongs to the group of humanoid robots; it is targeted for helping the elderly with difficulties of every day life; for instance it might be used for errands like bringing medicine or for video chatting with relatives. The robot has been put in environments like elder-care facilities.

2) Hardware: There are two computers installed on the platform. Firstly, there is a Linux machine which operates the hardware like motors and sensors. I do not have any access to this computer and from my point of view it is a black box. Secondly, there is a tablet PC HP Elitebook 2760p (Core i5 i5-2520M 2.5GHz, 4GB DDR3); the notebook runs Windows 7. It is a brain and face of the robot. Additionally,

thanks to the touch screen, it may be used as a user interface. The two computers are interconnected via standard Ethernet.

There are eight motors - two main motors for wheels, one to change angle of the camera, and five motors used in the robot's hand. It is also equipped with 10 ultrasound range sensors and frontal IR sensor for detecting the obstacles. Further, there is a stargazer to precisely determine the position of the robot.

Finally, there is a stereo-vision camera and Microsoft Kinect placed in the "head" part of the robot.

3) 3D vision and octree representation: ISRC develops the cognitive recognition system based on the 3D vision technology; the recognition system is utilized by the robot to make high-confidence decisions. Thus, the robot is equipped with 3D vision systems described above. The recognition system is based on octree representation of cells. For more information about the recognition system refer for example to [2], [3].

Currently, among other things, 3D SLAM for the robot is being created in ISRC.

B. Project description and motivation

In my term project for EME5721 I worked on 2D navigation in an unknown environment based on the vision. I used the Homemate to test the concept.

Homemate is already equipped with navigation software employing the integrated range finders and IR sensors; the navigation is based on a method using Dubin's curves. Yet, at the moment there are bugs which sometimes cause the robot to fail to reach the desired position. Therefore, I ventured to develop a navigation using the 3D vision described above.

Goal-directed navigation problem in an unknown environment is a standard task in Robotics. Many different solutions has been developed during the short lifetime of this field; this includes basic methods like Potential Field Method or Vector Field Histogram. In past two decades, with advances of computation power, methods based on grid-based search of robot's discrete map has become popular. Particularly with invention of the focused Dynamic A*, which allowed quick replanning, this kind of navigation became feasible for real time navigation [4].

In my project, for the path planning, I utilized an enhanced version of D*: D* lite [5].

Figure I.2. Basic environment

C. Environment

In our lab, we have a special spot dedicated for the robot. It is only a small $3\,\mathrm{m} \times 2\,\mathrm{m}$ rectangle; see Figure I.2. The spot is marked with signs for the stargazing sensor. Further, it is possible to move robot outside to the extended environment; see Figure III.1. When testing, I placed in the rectangle few artificial obstacles made of carton so the robot wasn't damaged due to errors in my algorithm.

II. Implementation

I chose to implement the path planning in C++ because the whole platform is developed using the language. Although, I have been extensively using C for simple applications this was my first encounter with C++. That was the first challenge while working on the project.

There were three major parts to deal with while programming the path planning. Obviously, the implementation of the D* lite algorithm itself was crucial. Next problem was obstacle detection and mapping. Finally, I had to solve how to move the robot. These problems are dealt with separately in the distinct classes. I describe the classes in following sections.

A. Obstacle detection and mapping

As I mentioned above, the 3D vision system was developed mainly with the recognition purpose. Each object in the field of view of the robot is represented with 3D cells using octree representation. This allows segmentation as well as recognition of the objects.

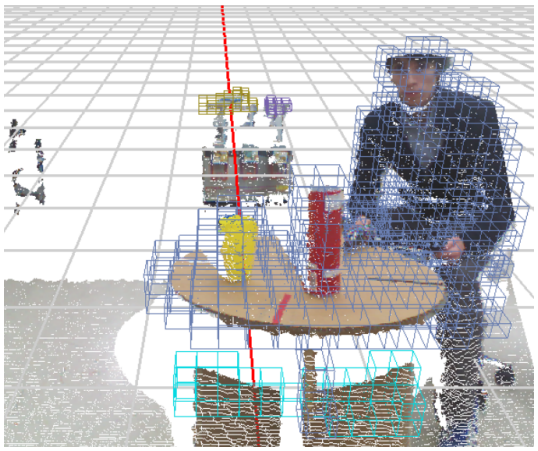

Figure II.1. Representation of 3D objects using octree

Yet, these functions are not important for the navigation. I am only concerned about the position of the octree cell which I match with the position of the obstacle.

Once the images from the stereo camera are processed, I have access to all the octree cells extracted from the image. Figure II.1 shows an example. The octree cells carry the information about the obstacle position in the coordinate system of the robot. I simply take all the octree cells and map them into 2D map using transformation:

$$\begin{aligned} \mathbf{x}_{o,g} &= \mathbf{x}_g + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \mathbf{x}_{o,r}\,, \\ \mathbf{x} &= \begin{bmatrix} x \\ y \end{bmatrix}, \end{aligned}$$

where $\mathbf{x}_g$ and θ are the coordinates of the robot in the global coordinate system, $\mathbf{x}_{o,g}$ is the position of the obstacle in the global coordinate system, and $\mathbf{x}_{o,r}$ is the position of the obstacle in the robot's coordinate system.

I must have dealt with the fact that the robot is larger than the cell. The solution was easy: when a cell is marked as an obstacle, neighboring cells within a certain range depending on the cell size are also marked as a unwalkable. I tested cell sizes 10 cm and 25 cm.

B. D* lite

The core of the whole navigation system is the D* lite algorithm. D* lite algorithm is very well described in detail in two publications by the original author [5], [6]. I use this paper as the main source for the implementation; I basically followed the pseudocode.

C. Robot movement

Once we know where the obstacles are and how to generate the path we can move the robot. Yet, this is not that smooth because I have access only to the limited number of functions to move the robot. Essentially, I can only move the robot forward and backward and rotate the robot; each action must be taken separately. Further, the robot control is based solely on the movement of the motors, there is no feedback from sensors or global localization system. One can imagine that this is not very precise method to control the robot.

The inaccuracy also limits the size of one cell; when a cell of the map is too small, the robot tends to move beyond the cell. At the end, I opted for cell size 10 cm which was a good balance between the traveled distance and achievable precision.

This limitation unfortunately also restricts me only on movement from one cell to the other instead of applying blended functions to smooth the trajectory while using a feedback controller to follow the trajectory.

I implemented a function which moves the robot from one map cell to the other. This method can be then called in a cycle to move to the next cell on the planned path.

Yet, movement from one cell to the other is very lagged since we must adjust rotation at every step. At least I tried to improve this behavior using a simple trick: if there is more successive cells in path aligned on a straight line, I move the robot to the end of the line. The maximum distance to move is limited to consider the limited range of the camera and thus to avoid collision with an obstacle.

D. User interface and navigation from start to goal

Using a console, user can see the map and position of the robot. Further, user can command the robot to go to a certain position on the map. In this case, D* lite planner is initiated to find the shortest path.

The robot moves from current cell to the next cell on a trajectory while mapping the environment. In case there is an obstacle, the planner updates the costs of

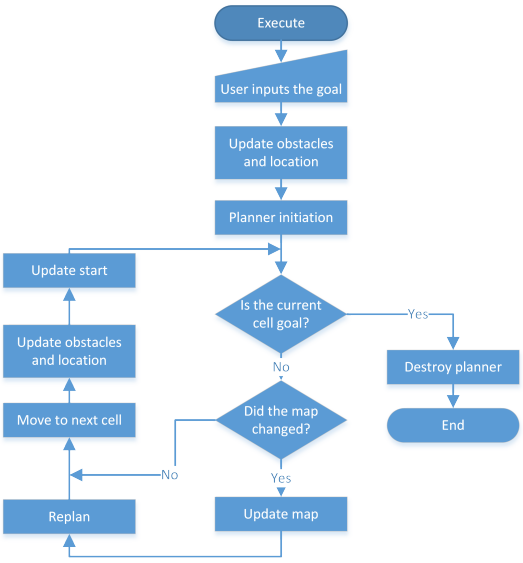

Figure II.2. Simplified diagram of planning

the cells and quickly replans. This repeats in a loop until the robot reaches the goal. If the planner fails to find the path, the robot gives up and ask the user to input a different location.

Refer to the diagram in Figure II.2 for a graphical illustration of the navigation.

III. Experiments

I performed several experiments which are captured in the included videos; you can find it online on [7].

First video shows the screen of the robot running the planning algorithm; on the left the console interface of the robot is shown. Further, on the right, there is the 3D reconstruction of image. I stand in front of the robot and move. The video demonstrates how the obstacle is mapped into 2D map and mainly it shows the performance of D* algorithm and its quick replanning.

Second and third video show the navigation of the robot in the simple environment described above. The robot is asked to go to the goal across the room. Then, the robot was supposed to return back. Homemate starts turned away so it doesn't have any information about the fact that there is an obstacle in front. In the second video, the grid size is 25 cm and in the third it is 10 cm.

In fact, the performance was worse for the second case for smaller grid size. The robot tended to go too

Figure III.1. Extended environment

far and thus it twice happened to go in the obstacle space. When this occured, the robot replanned the path. The problem was that now the optimal solution was to go in the shortest way from the obstacle space. It caused the problem that the robot turned perpendicularly away from the obstacle. For the larger grid size in the second video, this problem did not occur. On the other hand, the robot often goes on "zig-zag" path towards the goal.

In general, the robot was usually able to avoid the obstacle.

IV. Conclusion

In conclusion, in the project I used 3D camera to detect obstacles and navigate Homemate robot in an unknown environment. As I showed in the experiments, the robot is able to navigate using my D* lite algorithm in an unknown but simple environment from the current location to any position on the map if there is a possible path. The robot moves from cell to cell; with a small hack it can go faster when the line is straight. Obviously, this solution is far from the best but it was the most viable I could come up with given the restrictions on the control of the robot.

My solution likely won't replace the integrated navigation by Yujin. It wouldn't be plausible anyway to control the robot in the real time because of the delay from the Ethernet connection between the mainframe computer and the interface laptop.

I didn't have sufficient time to finish all the peculiarities of navigation. For example, it would be useful to implement "obstacle forgetting". Presently, the robot can navigate only in a static environment

and doesn't deal with moving obstacles. Further, it would be useful to thoroughly test the system and debug all the errors.

Acknowledgments

I would like to thank to Ahmed M. Naguib for letting me work with the robot despite there were deadlines to finish important parts of the recognition system. I am also grateful that he took his time to show me how to control the robot.

References

[1] 2013. [Online]. Available: http://inno.yujinrobot.com/ 1

[2] S. Lee, "Cognitive recognition and the homemate robot," in *2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2011, pp. 1–1. 2

[3] S. Lee, M. Ilyas, K. Jaewoong, and A. Naguib, "Evidence filtering in a sequence of images for recognition," in *2012 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 2012, pp. 1–8. 2

[4] A. Stentz, "The focussed d^* algorithm for real-time replanning," in *IJCAI*, vol. 95, 1995, pp. 1652–1659. 2

[5] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005. 2, 3

[6] ——, "Improved fast replanning for robot navigation in unknown terrain," in *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA '02*, vol. 1, 2002, pp. 968–975 vol.1. 3

[7] J. Tomasek, "Path planning for a mobile robot in an unknown environment using d* lite: Homemate robot," 2013. [Online]. Available: http://youtu.be/HxmRTQJ6JwY 4