Depth Image based Path Planning using the A* algorithm

Suvam Bag
Department of Computer Engineering
Rochester Institute of Technology
Rochester, USA
sb5124@rit.edu

***Abstract*—Traditionally mobile robots navigate with the help of depth sensors to avoid obstacles and build a path for itself. However depth sensors alone often prove to be insufficient in detecting obstacles at their specific locations because of their varying shapes and sizes, especially in unknown environments. Image based path planning can be very efficient for navigation of autonomous mobile robots in such kind of environments. Besides a normal image, a depth image can be even more beneficial for this purpose and it is also easily available from modern sensors like the Microsoft Kinect, Asus Xtion etc. A good path planning algorithm like the A* along with the help of these kind of sensors can be much more effective than normal autonomous robots using only distance sensors to build a map of the environment and navigate in it.**

I. INTRODUCTION

The paper proposes a unique path planning approach for mobile robots with the help of an area map created from a depth image in real time. The project was done on a relatively new platform called Robot Operating System (ROS).

The paper can be essentially divided into two main parts – (i) the navigation algorithm and (ii) mapping with the help of depth images captured in real time. The navigation algorithm used in this paper is a very famous algorithm often used in other domains like networking etc. It's called A*. The A* finds the shortest path between an initial position and a known goal position. This algorithm was used in this paper keeping in mind that autonomous mobile robots need to reach their goal position taking the shortest path possible to make it computationally efficient. This is one of the essential requirements of the mobile robots like the Mars Rovers.

The second part of the paper focuses on the use of depth images to learn the environment of the mobile robot in real time and use that to create a 2D grid map for navigation. The Asus Xtion was used for this purpose. The camera was placed on top of the mobile robot so that it can get a clear vision of the path in front of the robot (figure 1). The depth image taken at every instant was converted into a 2D map which could be used in the A* algorithm. Based on the calculated path, the robot moved to its goal position avoiding the obstacles in its path and take the next image and repeat the same, thus following a reiterative approach to make the algorithm dynamic.

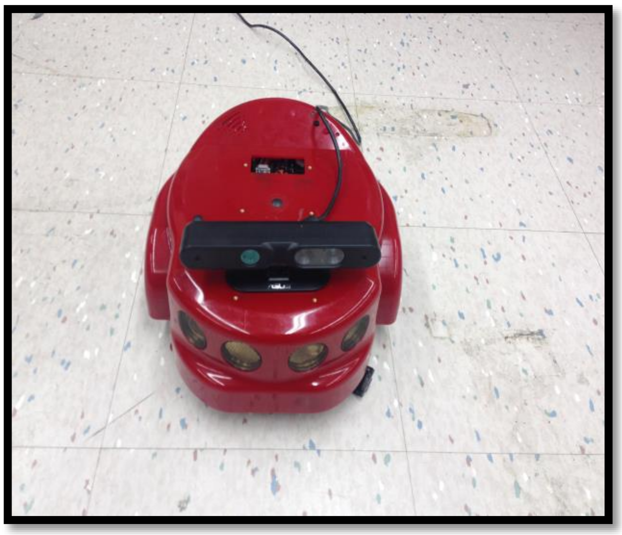


Figure 1: AmigoBot with the Asus Xtion placed on top

II. RELATED WORK

Navigation has been a subject of interest in mobile robotics over the past two decades. Despite a considerable amount of research been done in both academia as well as corporate level on autonomous robot navigation, various challenges and problems still remain. Navigation can be broadly divided into a lot of subareas like path planning, localization, mapping, Simultaneous Localization and Mapping (SLAM) etc. This paper primarily focuses on the path planning aspect of navigation using mapping techniques derived from depth image manipulation. Image based path planning applying image processing techniques have been proposed in the past [1] and in fact have been modified to a high level on the Mars Rovers. While some papers have proposed image processing filtering techniques like edge detection etc. for detecting obstacles and form maps, others have used machine learning techniques to learn the new environment from existing models [2]. Moving on to the depth images, the concept of point cloud comes into the picture. Despite modern sensors giving the point cloud data, it is very difficult to create the map out of that data. Moreover, the sheer amount of data provided by the depth images makes the computational complexity check one of the top priorities in autonomous navigation. Innovative algorithms to reduce this huge volume of data have been proposed in several research papers like the Fast Sampling Plane Filtering [3]. A number of different path planning algorithms have been proposed in the past and [4] gives a good review of some of them. SLAM is another hot topic of research in the field of autonomous robot navigation. Abundant research is going on in this field, using different sensors like the Kinect [5], [6], [7], [8]. The Kinect has been often used in SLAM, because it is the one of the best pre-calibrated camera capable of giving a 3D scan of the environment. Ever since the invention of ROS, the research in this field has improved a lot benefiting from the open source community. ROS packages are available now which aid in building SLAM algorithms. [9][10] The Kinect which is very similar to the Asus Xtion used in this paper has been used extensively in many indoor navigation projects in the past because of its rich resources implanting different algorithms along the way for better path planning. [11]

III. Platform and Sensors

A. Robot

The mobile robot used in this paper was the AmigoBot from adept mobilerobots. It is a small cost-effective, differential-drive robot primarily used for education and collaboration projects. The AmigoBot features 8 sonar sensors although they haven't been used in this paper and can be operated in tethered mode over wirelessly with the optional communications necessary. A dedicated ROS library for this robot known as the RosAria was used in this paper.

B. Asus Xtion

The Asus Xtion is very similar to the Microsoft Kinect used abundantly in various papers. The advantage of using this over the Microsoft Kinect is that it doesn't need an external source of power. Hence it was easier to use for autonomous navigation. It is also capable for providing the depth image of the environment at different resolutions at different frames per second. The OpenNI2 is a dedicated tool for using this device and for performing the image processing and computer vision techniques on the acquired image, OpenCV2 was used.

C. ROS

The operating system used was ROS-Indigo on Ubuntu and the program was developed in Python. Besides the mentioned libraries and tools, no other packages or tools were used in this paper. For a good introduction to ROS, readers are encouraged to review the tutorials of ROS-Wiki.

IV. Theory

The paper can be broadly divided into two sections –
(i) Use the depth image for mapping and (ii) the A* algorithm for navigation in a 2D grid created from the depth image.

A. Acquiring depth image

Figure 2 shows the RGB image obtained from the Xtion's RGB sensor. This image is displayed only for the purpose of showing the view of the Xtion Sensor on the robot and thus eases the task of comparing the depth sensor's image as well as the processed images.

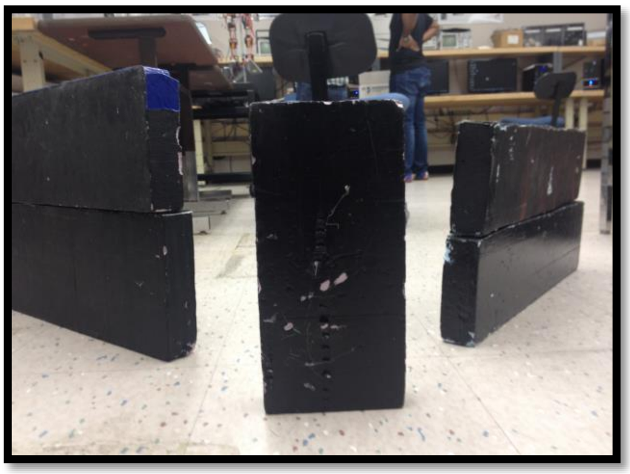


Figure 2: RGB image of obstacles in the robot's path

Firstly, the 11-bit raw data (which is an array of 640x480 elements) was retrieved from the Xtion. Then this image was cropped to 640x240 resolution. This was done because the robot's range of view only needs to focus within the height of the robot and after several calibrations, the resolution was determined to be 640x240 with respect to keeping the camera at a specific angle on top of the robot. From the depth image in Figure 3, it can be seen that, darker the pixels, the nearer the object is. The lighter pixels belong to the wall in the background. An alternative visualization of the depth image taken from the Rviz has also been shown in Figure 4.

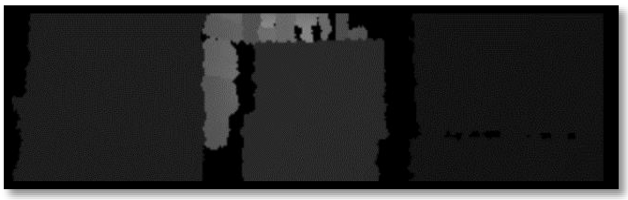


Figure 3: Cropped 640x240 depth image

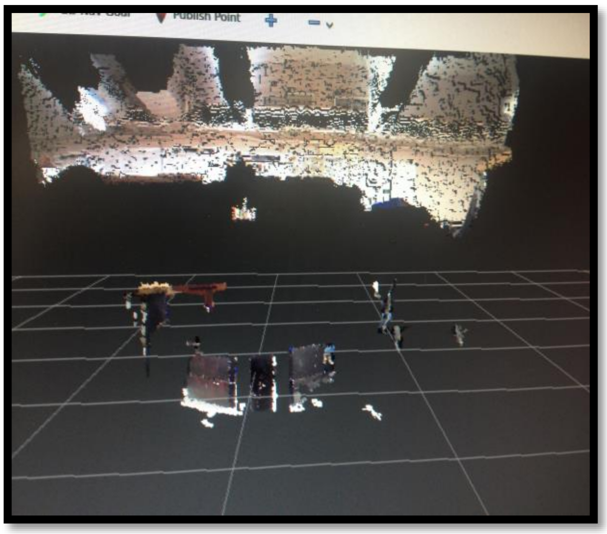


Figure 4: Depth image from Rviz

B. Mapping using the depth image

To create a 2D map out of the acquired depth image, the depth pixels need to be converted to a 2D coordinates. Hence the next step was to calculate the X and Y coordinate that correspond to each depth pixel (i.e. Z coordinate) were calculated using

$$\mathbf{X}_{i,j} = (\mathbf{j} - \tfrac{w}{2}) \times \frac{320}{w} \times \mathbf{M} \times \mathbf{Z}_{i,j} \qquad (1)$$

$$\mathbf{Y}_{i,j} = (\mathbf{i} - \tfrac{w}{2}) \times \frac{120}{w} \times \mathbf{M} \times \mathbf{Z}_{i,j} \qquad (2)$$

where i and j are the pixel's row and column number in the Z-array respectively, w and h are the width and the height of the Z-array and M is the NUI Camera Depth Image to Skeleton Multiplier Constant. These equations were obtained from the Microsoft SDK library for Kinect and the value of M was taken as 0.0021.

Now there are three arrays of 640x240 pixels representing the real X-Y-Z coordinate. Next the minimum element in each column of the Z-array was selected such that

$$\mathbf{Z'}_{j} = \min\left(\mathbf{Z}_{0,j}, \mathbf{Z}_{1,j} \ldots \mathbf{Z}_{239,j}\right) \qquad (3)$$

where j is the respective column number. The X locations for the corresponding Z-elements were also selected giving another 640x1 array. These two arrays indicate the Z and X coordinates of the nearest pixels and can also be regarded as the 2D obstacle locations. The minimum-selection method in Eq. 3 was implemented to avoid false detection of obstacle and minimize processing power and time. The Y-Coordinate was ignored as the robot can only move in X and Z directions. The vertical heights of the obstacles are irrelevant in this case as the robot only needs the width and the distance from itself to create the 2D grid map. The obstacles from the RGB image from Figure 2 were mapped using this logic and is shown in Figure 5. Note that the location of the obstacles are relative to the Xtion's point of view. The wall behind the obstacles is not visible since only the nearest obstacles were mapped (refer Eq. 3). The obstacles behind the mapped ones were avoided because an unseen object might have been present between those and the wall. From this map a 2D grid was created and sent as an input to the A* algorithm.

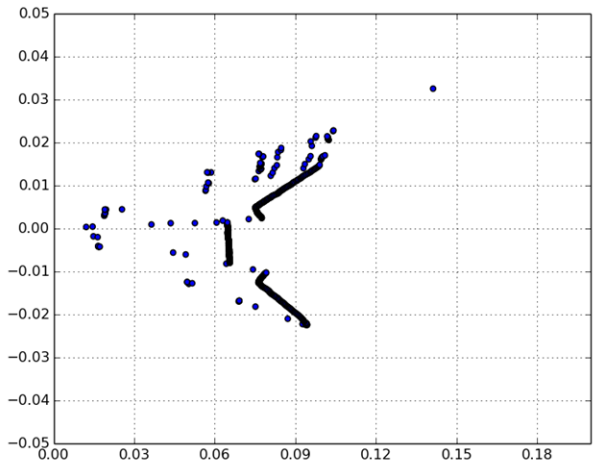


Figure 5: 2D map created out of the X-Z coordinates

B. The A algorithm*

The A* algorithm is a very famous algorithm often used in navigation of autonomous robots and even other domains also like networking to find the best route between two stations etc. It is a corollary of the even more famous Dijkstra's algorithm. Basically it is used to find the shortest path between two nodes or stations with the help of a cost function and a heuristic function. A* requires a distance function from start configuration to goal configuration. It does not need an exact distance, just an estimate of how long it would take to get from start configuration to the goal configuration, in the best case.

A* uses a best-first search and finds a least-cost path from a given initial node to one goal node (out of one or more possible goals). As A* traverses the graph, it follows a path of the lowest expected total cost or distance, keeping a sorted priority queue of alternate path segments along the way. It uses a knowledge-plus-heuristic cost function of node x (usually denoted f(x)) to determine the order in which the search visited nodes in the tree. The cost function is a sum of two functions:

- The past path-cost function, which is the known distance from the starting node to the current node x (usually denoted g(x))
- A future path-cost function, which is an admissible "heuristic estimate" of the distance from x to the goal (usually denoted h(x)).

In this paper, the A* algorithm was slightly revised so that instead of just finding the length of the shortest path, it also recorded the actual sequence of steps. Each node on the path kept track of its predecessor. After running the full algorithm, the end node pointed to its predecessor and son on until some node's predecessor was the initial start node.

In this paper, the A* algorithm was implemented with respect to the intended navigation of the mobile robot. Four actions – left, right, down and forward were given as velocity commands. For travelling in the 2D grid map, the cost function was created as [2, 1, 1] corresponding to left, forward and right turn. The weight corresponding to turning left was given a higher value because in the US system, where vehicles travel on the right side of the road, it is difficult to turn left than turning right, Figure 6. Based on this fact, the mobile robot should always try to calculate its path keeping the cost minimum and always a choosing a right turn over a left turn whenever possible. The ultimate goal is to find the shortest path keeping the total cost minimum. The algorithm was also made so that it can take 3D input and form the 2D grid map. This added to the already efficient algorithm, because now it could also take the orientation as a parameter or some other third parameter if required.

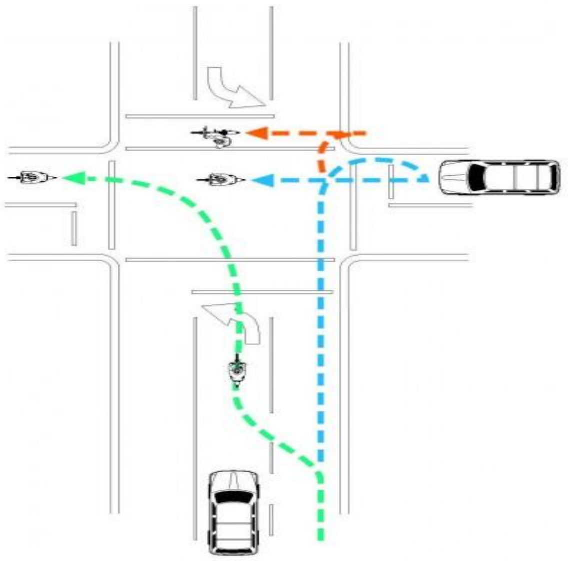


Figure 6: Left turn vs right turn for normal cars

C. Main algorithm

Combining sections A and B and setting up in the ROS platform is important to make the robot function properly. After subscribing to the depth image and then publishing it, the necessary processing of depth pixels was done to be able to convert it to a 2D grid map. This was transferred to the A* algorithm which calculated the best path along with the appropriate velocity commands. Based on them, the command velocities were published to the RosAria library of the mobile robot. After reaching to the goal position of the 2D grid acquired from the first depth image, one more image was taken and the 2D grid was calculated again, thus making the whole process dynamic. It should be noted that the central node unsubscribed from the camera/depth/image topic after getting the first frame every time and only re-subscribed after completing the first grid using the A* algorithm. The algorithm is better explained in Figure 7.

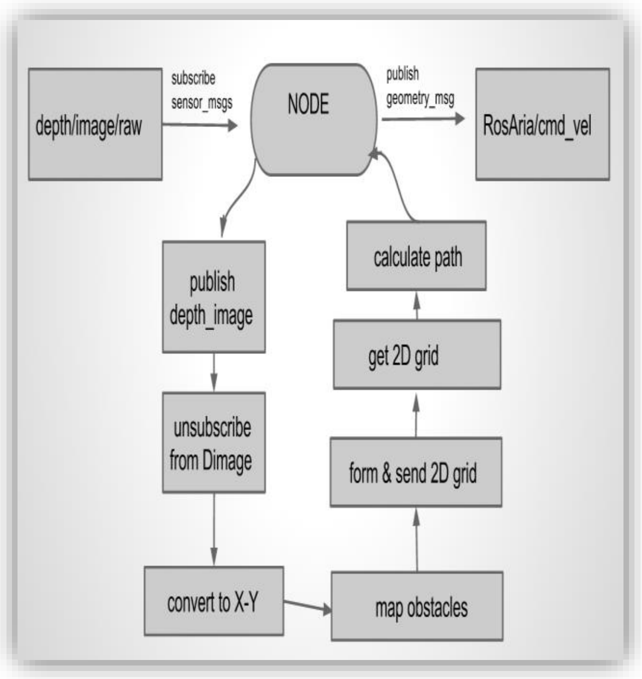
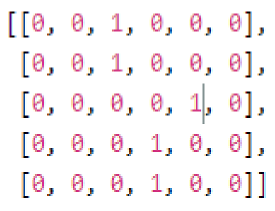


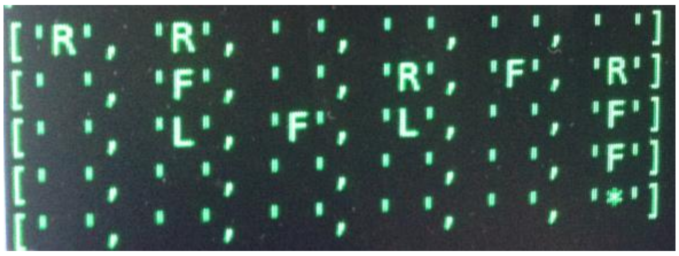
Figure 7: Main algorithm in ROS

V. Results

Several trials were conducted both for the mapping and the navigation and the mobile robot based on that before concluding the paper. In most of the cases the obstacles were mapped correctly on the 2D map. The mobile robot was also able to reach the end goal on the grid based on both user defined positions and predefined goal positions on the map in such cases. A 2D grid map has been shown in Figure 8(a), where 0's are free space for allowable for propagation and 1's are grid cells occupied by obstacles. The calculated path from the algorithm has also been shown in Figure 8(b).


```
[[0, 0, 1, 0, 0, 0],
 [0, 0, 1, 0, 0, 0],
 [0, 0, 0, 0, 1, 0],
 [0, 0, 0, 1, 0, 0],
 [0, 0, 0, 1, 0, 0]]
```

(a)



```
['R', 'R', ' ', ' ', ' ', ' ']
[' ', 'F', ' ', 'R', 'F', 'R']
[' ', 'L', 'F', 'L', ' ', 'F']
[' ', ' ', ' ', ' ', ' ', 'F']
[' ', ' ', ' ', ' ', ' ', '*']
[' ', ' ', ' ', ' ', ' ',
```

(b)

Figure 8: (a) 5x6 2D grid map, (b) calculated path by the A* algorithm on the 2D map in (a) showing the move directions from the (1, 1) grid cell as initial position to goal position '*'

VI. Conclusion

In most of the cases, the algorithm worked perfectly and the robot was able to reach the goal position from the grid map, Figure 9. The algorithm was also made flexible so that it could take 3D coordinates for building the map. Hence height of the obstacles can also be mapped if desired. But since in this case, the robot had negligible road allowance, hence the height was not considered since ever obstacle were obstacles on its path. However since the paper has minimized the depth pixels to consider the nearest obstacles, image occlusion is a scope of improvement for this paper. This is because, objects blocked by objects in front of them, and could not be mapped. It was assumed in this paper, that there were no obstacles behind the ones mapped in Figure 8(a).

References

[1] H. N. Joshi, Prof J. P. Shinde, "An Image Based Path Planning Using A – Star Algorithm," International Journal of Emerging Research in Management & Technology ISSN: 2278-9359 (Volume-3, Issue-5)

[2] G. N. Tripathi and V. Rihani, (2012) "Motion Planning of an Autonomous Mobile Robot using Artificial Neural Network," *CoRR* **abs/1207.4931**

[3] J. Biswas and M. Veloso, "Depth Camera Based Indoor Mobile Robot Localization and Navigation," 2012 IEEE Intrenational Conference on Robotics and Automation. RiverCentre, Saint Paul, Minnesota, USA, May 14-18, 2012

[4] N. Sariff and N. Buniyamin, "An Overview of Autonomous Mobile Robot Path Planning Algorithms," 4th Student Conference on Research and Development (Scored 2006), June 2006.

[5] T. Emter and A. Stein, "Simultaneous Localization and Mapping with the Kinect sensor," Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on 21-22 May 2012, Munich, Germany

[6] J. Hartmann, D. Forouher, M. Litza, J. H. Klussendorff and E. Maehle, "Real-Time Visual SLAM using FastSLAM and the Microsoft Kinect Camera," Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on 21-22 May 2012, Munich, Germany

[7] Y-T Wang, C-An Shen and Jr-Syu Yang, "Calibrated Kinect Sensors for Robot Simultaneous Localization and Mapping," Methods and Models in Automation and Robotics (MMAR), 2012 19th International Conference on 2-5 Sept. 2014, Miedzyzdroje

[8] H. Jo, S Jo, E. Kim, C. Yoon and S. Jun, "3D FastSLAM Algorithm with Kinect Sensor," Soft Computing and Intelligent Systems (SCIS), 2014 Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on 3-6 Dec. 2014, Kitakyushu

[9] J. M. Santos, D. Portugal R. P. Rocha, "An Evaluation of 2D SLAM Techniques Available in Robot Operating System," Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on 21-26 Oct. 2013, Linkoping

[10] S. Gong, H. Liu, Y. Hu, J. Zhang, "ROS-based object localization Using RFID and laser scan," Information and Automation (ICIA), 2012, International Conference on 6-8 June 2012

[11] Y. Zhou, G. Jiang, G. Xu, X. Wu, L. Krundel, "Kinect Depth Based Door Detection for Autonomous Indoor Navigation, " The 23rd IEEE International Symposium on Robot and Human Interactive Communication August 25-29, 2014. Edinburgh, Scotland, UK

Appendix

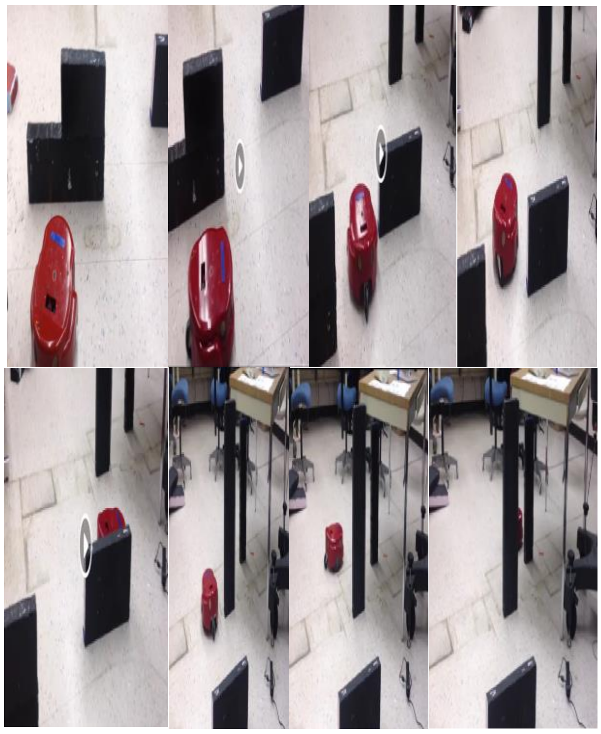


Figure 9: Real time implementation