Navigating YouBot through a rose field with A*

Jasper van Enk

6150519

Bachelor thesis

Credits: 3 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam

Faculty of Science

Science Park 904

1098 XH Amsterdam

Supervisor

dhr. dr. A. Visser

July 26th, 2013

Abstract

This article will describe the first step of three steps to be taken in this YouBot rose field project. Although it does not yet take into account unexpected deviations from its path when limited to the use of the A* algorithm alone, A* does prove to be a sufficient path planning algorithm for the task of navigation YouBot through a field of roses, while driving by every one of them. A short piece of video footage of this project on youtube gives an impression of the YouBot's performance.

Contents

Introduction

Recent developments show the great potential robotics have to play a significant role in agricultural activities (IEEE Robotics and Automation Society Technical Committee on Agricultural Robotics and Automation).

The purpose of this research is to gain insight in the performance of path finding algorithm A* when applied to the YouBot in the "rose field task". This article will describe the first step of three steps to be taken in this YouBot rose field project. This first step will enable the YouBot to navigate through a "regular" rose field consisting of neat rows. The second step will be to navigate through an "irregular" rose field consisting of messy rows that possible even force the YouBot to choose an alternative path. Lastly, the third step will enable the YouBot to actually search for a specific rose and pick it up. Since the YouBot is well equipped, several approaches to the problem are possible, of which one is described in this article. The software created in these three steps should enable the YouBot to take part in the KUKA challenge. This three step project will be "UvA @ Work" team's admission in the KUKA challenge.

In this first step of the project, a path finding algorithm will be used to plan a path through a predetermined grid. This grid contains roses, which all have to be visited (YouBot has to drive by them) before going to the goal position in the grid.

Literature Review

A recent review on the developments in agriculture tasks automation (Bergeman, van Henten, Bilingsley, Reid, & Mingcong, 2013, June) displays a dramatic increase in research papers on robotics in agriculture in the past 10 years. Highly advanced robots are now able to perform automated weeding, thinning, plant classification and many more tasks. Robots have until today been proven to lack the skill in agricultural tasks of humans like harvesting. Other, simpler, actions (e.g. pruning and thinning) however show that robots have very high potential to contribute significantly to the future of agriculture. The RoseRunner (Hafren, et al., 2012) project is a striking example of these simpler tasks being performed by a robot. Its conclusion states however:

> "*The main considerations now, when the project is near to closing, are that the robot should be simpler and more attention should be paid to the artificial intelligence and less to the secondary subsystems."* (Hafren, et al., 2012)

This is the motivation for this project to focus on exactly what is pointed out in the conclusion of the RoseRunner project. A relatively "lightweight" path finding algorithm will be implemented to display what it will make the YouBot capable of, and maybe even more importantly, what it does not make the YouBot capable of.

The inspiring article about detection of topological structure from sensor data (Dolgov & Thrun, 2009) will be one of the long-term goals for the YouBot to adopt. Both the approach and application (mapping and driving through a semi-structured environment) described in the article are very compatible with the task described in this article for YouBot in the rose field.

While this article focusses on a simple solution and its abilities and limitations for reasons described earlier, more complex alternatives are important to keep in mind. Rapidly exploring random trees (RRT) is an approach described in "Motion Planning" (Kavraki & LaValle) which is one of those alternatives.

Material & Method

A KUKA YouBot with the following specifications was used for this project:

Figure 1: Kuka youBot equipped with Asus Xtion Pro Live camera, Xsens MTi-G-700 sensor and Hokuyo LX30 laserscanner

To be able to measure the performance of the YouBot using an A* path planner, first an A* implementation was required. This implementation was created in JAVA using Wikipedia [1]pseudo code. While the YouBot runs on a C++ API, it was configured by (Negrijn, 2013) as a server that is able to receive "strings" that carry commands for the YouBot. Therefore this setup allows the user to send strings of a certain composition from any kind of programming language through a client to the YouBot server. JAVA software for the YouBot to be able to drive the path found by the A* algorithm has been developed supported by JAVA classes (e.g. "NetworkClient") created by (Negrijn, 2013).

Second, a sufficient rose field is required. Potted roses were arranged in rows of three as displayed in figure. The roses are 17.5 by 17.5 cm and placed along each other by a distance of approximately 20 cm. Each row has a distance of approximately .75 m to the next. Since the test environment contains three rows of roses, YouBot is required to drive through its 5 "lanes".

Figure 2: Rose Field, each white plate represents a node which the YouBot is not allowed to go to.

[1] http://en.wikipedia.org/wiki/A*_algorithm

Software

The software architecture created for this project, displayed in figure 3 resembles the one used in the application made by (Negrijn, Erich, Wandenaar, & Koster, 2013) in the way the packages are assembled. However, the core of the software used in this project is the Planner package and the main class ("YouBotController") which have been created specifically for this A* program. Not all classes are used in this specific application because they enable the use of the arm that the YouBot is equipped with. In this step of the project, the arm is not used yet.

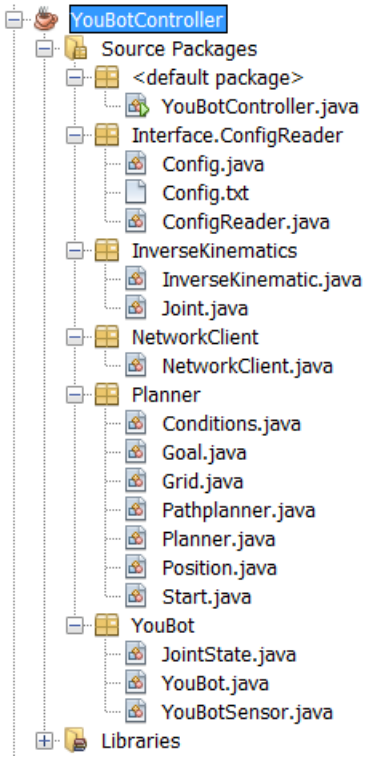

Figure 3: Software Architecture

Planner

The "Planner" package contains a planner class that converts a path that is found by A* into driving orders from node to node for the YouBot. These driving orders are sent one by one through the Client to the YouBot Server. The planner class receives its input from the "Pathplanner" class which is where the A* algorithm is executed given a set grid.

YouBot

Since the YouBot class was originally part of the program created by (Negrijn, Erich, Wandenaar, & Koster, 2013), it was focused on constructing orders for the YouBot to move its arm (using the "Joint", "JointState" and "InverseKinematic" classes). The "YouBot" class was rebuild to focus mainly on movement in order for the A* path to be driven by the YouBot. The "Planner" class is now able to output Strings to through the Client to the server of YouBot using the "YouBot" class (e.g. setDrive() method).

The Grid, Goal, Start and Position class represent objects with corresponding names that are used throughout the rest of the program.

Data

The grid representing the real world of the rose field was declared as a 2-dimentional array of coordinates of x and y, each containing 0 or 1. For each node in this array, 0 means it is accessible for the YouBot, and 1 means it is not (because a rose is on that node). Each rose was represented as being one node on the grid, meaning that when the YouBot is driving straight, it passes a rose with each node it drives through. A* uses the Position objects (corresponding to grid coordinates) to assemble an ArrayList of Positions that will form the path to the goal.

A*

The A* implementation uses the common approach of keeping an open nodes list and a closed nodes list to keep track of whether or not A* has already visited a node. A* continues searching until the goal is reached, or until the open nodes list is empty (meaning that there are no nodes left to go to). In the latter case, it means that A* has finished its search without reaching the goal, meaning that there is no path to the goal. Figures 4 and 5 show an example of both cases, in which A* behaves as to be expected. In each table the nodes are represented by squares. Each square containing a 1 is a node that cannot be accessed because a rose is standing there. Each square containing a 0 is a node that the YouBot is able to stand on, but has not yet done so. And finally, each square containing an x is a node that the YouBot has been to. The “s” and “g” represent the start and goal.

s	1	x	x	X	1	g
x	1	x	1	X	1	x
x	1	x	1	X	1	x
x	1	x	1	X	1	x
x	1	x	1	X	1	x
x	1	x	1	X	1	x
x	1	x	1	X	1	x
x	1	x	1	X	1	x
x	x	x	1	X	x	x

Figure 4: simple maze with possibility of reaching goal

s	1	x	1	0	1	g
x	1	x	1	0	1	0
x	1	x	1	0	1	0
x	1	x	1	0	1	0
x	1	x	1	0	1	0
x	1	x	1	0	1	0
x	1	x	1	0	1	0
x	1	x	1	0	1	0
x	x	x	1	0	0	0

Figure 5: simple maze without possibility of reaching goal

In figure 4 only one path is possible for reaching the goal, and A* has no problem finding it in less than a second. Figure 5 shows a maze in which no path is possible to reach the goal. A* still finds a path that

reaches halfway, but then terminates and states that it could not find a path, because it has ran out of nodes in the open nodes list.
A* has no trouble if the mazes increase in difficulty. It finds the shortest path to the goal in the maze displayed in figure 6 in less than a second. If a path is found between the starting and goal position, and it is compared with the path that is found when the original goal and starting position are swapped, A* finds exactly the same path as it did before the swap.

s	x	x	1	0	1	x	x	x	1	x	x	x	x
0	1	x	x	X	x	x	1	x	x	x	1	0	x
1	1	x	1	1	0	1	1	0	1	1	0	0	x
x	x	X	1	0	0	1	0	0	1	0	0	1	x
0	1	x	1	0	1	0	1	0	1	0	1	0	x
1	1	x	1	0	0	1	1	0	1	0	0	0	x
x	x	X	1	1	1	0	0	0	1	1	1	0	x
x	x	1	1	x	x	x	x	1	1	x	x	x	x
x	1	x	x	x	1	1	1	x	x	X	1	1	x
x	x	X	1	x	x	x	x	X	1	x	x	x	g

Figure 6: complex maze

Server

To be able to receive strings containing commands, the YouBot uses the YouBotNetwork application in c++ which can be found among the provided code of the YouBot. Each string starts with the letters DRIVE, STOP or JOINT, making clear what category the command is in. The letters are followed by numbers, depending on what letters are in the command. The JOINT keyword requires a number for each degree of freedom, the DRIVE keyword and STOP keyword not require numbers **(CHECK THIS)**.

Results

Mazes of different quality show that the path finding problem of the rose field can be easily handled by the A* algorithm. YouBot is successfully lead from node to node in the virtual grid presented in java code, resulting in an almost equally successful real world performance.

YouBot drives through the rose field successfully without collisions. However, factors like lack of grip on the floor can cause differences between the virtual position and the real world position the YouBot is in. The YouBot is not able to compare its virtual world position with its real world position. Furthermore, YouBot has some trouble taking into account its own length, meaning that the “Planner” class that takes the path from A* and leads YouBot from node to node, is actually required to recognize a situation in which the YouBot should make a turn and then (while not prescribed by A*) has order YouBot to drive on for a few seconds before actually taking the turn.

A short piece of video footage of this project on youtube[2] gives an impression of the YouBot’s performance. In this video the YouBot holds still for one second before driving to each next node. This was done in the video for the purpose of showing the nodes that the YouBot drives through. In some cases YouBot holds still for more than one second, which is caused by an unstable connection between

[2] http://youtu.be/BuJbjpK1QU8

the server and client. The command that was sent to the YouBot is not received the first time, after which the client is instructed to wait for a little while and then try to send the command again.

An inconvenience of the YouBot is that it must be told how many seconds it has to drive in a specified direction. When distances between nodes are less than a second, the YouBot has no other option but to lower its speed to be able to meet the required distance.

Discussion

Although it does not yet take into account unexpected deviations from its path when limited to the use of the A* algorithm alone, A* does prove to be a sufficient path planning algorithm for this task. A* leads the YouBot from its starting position along the nodes to its goal position collision free and along the shortest possible route while driving by every rose. For the A* algorithm to work, a simple adjustment was made in representing the rose field map, to make it look like there was only one path possible. In step two of the project objects can be found along the way blocking the route to the goal. A* can create a new goal position (temporarily forgetting the actual goal position) to which it has to drive first (to make sure YouBot still drives by every rose in the field). When that new goal is reached, the original goal can be placed back, and YouBot can continue along its route.

YouBot should be able to recognize corners in the maze and take into account its own length when making a turn. In the implementation used in this project the length of the YouBot is compensated in the way that YouBot drives a tiny bit further when driving to a next node than necessary. This means that at the end of the row, YouBot will have driven its own length extra on top of the distance to each next node.

References

Bergeman, M., van Henten, E., Bilingsley, J., Reid, J., & Mingcong, D. (2013, June). Society Technical Committee on Agricultural Robotics and Automation. *IEEE Robotics and Automation*, 20-23.

Dolgov, D., & Thrun, S. (2009). *Autonomous Driving in Semi-Structured Environments: Mapping and Planning.* Kobe, Japan: 2009 IEEE International Conference on Robotics and Automation.

Hafren, J., Alaiso, S., Karppanen, E., Koskinen, M., Kostiainen, J., Rannisto, J., . . . Valli, A. (2012). *RoseRunner.* Venlo, Netherlands: 10th Field Robot Event 2012.

Kavraki, L., & LaValle, S. (n.d.). Motion Planning. *Kavraki_chapter5*.

Negrijn, S., Erich, V., Wandenaar, M., & Koster, J. (2013). *Basic order picking met behulp van de KUKA YouBot.*