AgriBOT PROJECT – COMPARISON BETWEEN THE D* AND FOCUSSED D* NAVIGATION ALGORITHMS

Gabriel Queiroz Silva Abrahão, gabrielabrahao@yahoo.com.br
Poliane Torres Megda, politorresmegda@hotmail.com
Henry Borrero Guerrero, borrelo@gmail.com
Marcelo Becker, becker@sc.usp.br

University of São Paulo – Engineering School of São Carlos - Department of Mechanical Engineering, Mechatronics Group, Mobile Robotics Laboratory (USP – EESC – SEM – GrMk - LabRoM), Av. do Trabalhador são-carlense, 400, Pq. Arnold Schmidt, São Carlos - SP - Brazil - CEP 13566-590

Abstract. *The use of robotic systems, especially mobile robots, in farming applications has spread worldwide since the early 1990s. Today this application is widely known as precision agriculture. Several research groups have been focusing their studies in such application, aiming the reduction of production costs for agricultural products. The AgriBOT Project is a cooperation project between the Mechanical Engineering Department at Engineering School of São Carlos (SEM - EESC - USP), EMBRAPA, and Jacto Company, in development at USP São Carlos. This project aims to develop an agricultural mobile robot able to navigate autonomously in a crop, without damaging the plants, and collect data and samples. This paper presents 2 different algorithms applied to plan navigation routes for a mobile robot: D* and Focussed D* algorithms. The first one is an algorithm that allows a mobile robot to move to a predetermined destination based on its initial and desired positions on the map of the environment. It is designed to operate in unknown environments. The second algorithm is the Focussed D* algorithm that is a version of the D* algorithm designed to use maps that are incomplete or may have many errors (e.g.: the presence of unmapped obstacles). We observed that in situations where the environment map is static, such as plantation fields, the D* algorithm can be used to trace the route of the agricultural robot requiring a lower computational cost when compared with more sophisticated algorithms that deal with complex and dynamic environments. If the environment is a dynamic one, the algorithm becomes less efficient, because it considers the moving obstacles as errors in the map (new obstacles that appear in the environment). The performances of the algorithms are compared in terms of the time required to generate the paths between the initial and goal positions.*

Keywords*: Agricultural robotics, mobile robot, path planning algorithms, D* algorithm, Focussed D* algorithm.*

1. INTRODUCTION

Recently, the technological advances in the various knowledge fields have allowed the adaptation of techniques developed for outdoor mobile robotics in the field of agriculture, increasing profitability and reducing environmental impacts. There is a great mobilization of research groups in this area, mainly due to the need of increasing food production and reducing costs. Today, a keyword in this area is widely known as precision agriculture that encompasses not only the automation of the processes, but also the planting data acquisition, detection and diagnosis of problems (pests, plants malformation, etc.), and customized treatment.

In this scenario, mobile robotics can be applied in agriculture to optimize some laborious activities, e.g.: harvesting, spraying, seeding, and soil preparation. Some interesting examples of mobile robots developed for application in agricultural fields can be found in (UNIBOTS, 2011). However, many challenges must be overcome in order to become the use of robots economically viable in agriculture. A good example of challenge is the robot navigation system. An agricultural mobile robot needs an onboard path planner that calculates a collision-free trajectory sufficiently precise that enables the robot to execute its tasks without damaging the crop.

The basic idea of a path planner is to find a number of states that allows the robot moving from a start position to a goal position without colliding with obstacles. So, an optimal path is a sequence of robot states, where the sum of the crossing costs has a minimum value. In order to plan a path, it is necessary to use a map of the environment (a local or a global map). This map shows the positions of known obstacles (static obstacles). By using the map data one can also detect mobile obstacles and use these data to obtain a collision-free optimal path. When it comes to a plantation, it is highly probable that the environment can be classified as static. So, the possible obstacles and the boundary conditions are well known and controlled. The algorithms that deal with this kind of situation are simple, fast, and computationally efficient. But, for safety reasons, it is necessary to avoid as much as possible risk of accidents. Due to this, even if the robot need only GPS, IMU, and odometry data to perform a predefined path in a plantation, more sensors need to be added to monitor the robot surrounds and avoid accidents (e.g.: running over, fatalities, etc.). Sensors like laser rangefinders, sonars, and cameras are common choices to execute this task in dynamic enviroments. In addition to this, it is necessary to highlight that algorithms that deal with such kind of data are more complex, requiring more embedded processing power, but they are more reliable.

In the literature there are several algorithms that compute a trajectory from a starting point to a goal position in a static and known environment. The A* algorithm was introduced by Nilsson (1980), while the Field A* algorithm (Abrahão et al., 2010) is an adaptation of the Field D* algorithm, which was created by Ferguson and Stentz (2005-a). In (Abrahão et al., 2010) we present some results concerning the implementation of both algorithms and we compared

their performance. The D* algorithm, presented in Stentz (1994), is an algorithm that calculates routes in unknown static maps. In addition to this, the Focussed D*, proposed by Ferguson and Stentz (2005-b), present a new replanning algorithm that generates equivalent paths to D* algorithm while requiring about half its computation time. Like D* algorithm, it incrementally repairs previous paths and focuses them towards the current robot position. In this paper we present the D* and Focussed D* algorithms applied in planning collision-free path for an agricultural mobile robot: the AgriBOT. The paper is divided as follows; section 2 presents in a nutshell the AgriBOT project. In the third section we introduce both algorithms: the D* and the Focussed D* algorithm. The results are presented in section 4 and the conclusions, in section 5.

2. AgriBOT Project

The AgriBOT (Agricultural Robot) project is a cooperation project between EMBRAPA (Empresa Brasileira de Pesquisa Agropecuária), EESC – USP (Escola de Engenharia de São Carlos – Universidade de São Paulo), and Jacto Company with finantial support of FINEP (Financiadora de Estudos e Projetos), CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior).

(a) (b)

Figure 1: AgriBOT prototype. In (a) a Pro-Engineer drawing and in (b) a photo of the robot when it was delivered by Jacto Company in December, 2010.

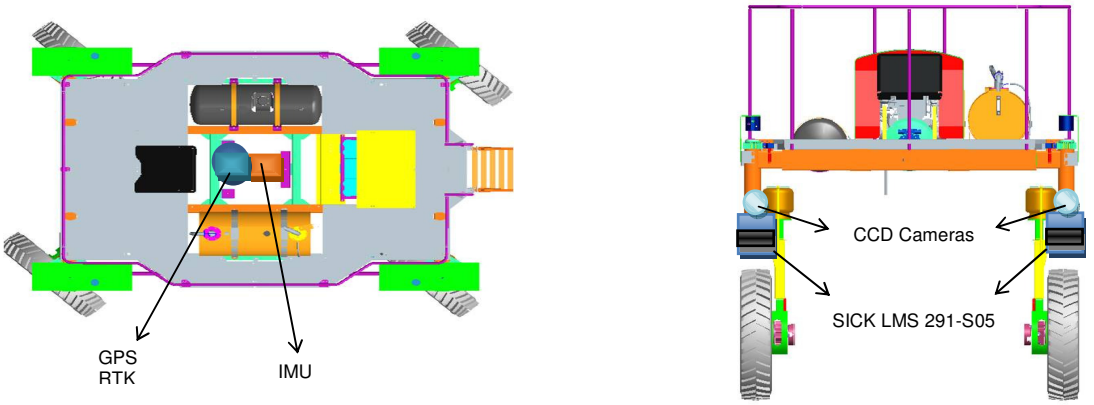
The AgriBOT mobile robot prototype consists basically in a robotic autonomous real scale four-wheel tractor endowed with the independent steering 4WSD configuration (a four-wheel steering and driven vehicle). In Fig. 1-a one can visualize a Pro-Engineer drawing of the mobile robot platform and in Fig. 1-b, a photo of the same robot when it was delivered by Jacto Company in December, 2010.

The basis for the robotic platform is the mobility capability provided by four wheel module mechanism that is controlled by an electromechanical interface. This interface includes its respective power driver and a single controller area network bus (CAN). Each one of the four identical wheel modules includes hydraulic motor actuators for propulsion system that provides direct drive. The steering capability is achieved by a separate steering hydraulic motor assembled on top of the wheel module shaft in order to create a two-degree-of-freedom system. The electronics control (wheel node) is based on a commercial agricultural job computer and handles the local position servo control for the steering. It also provides torque control of the driven motors. The driven motor electronics allows speed and current (torque) feedback while the steering servo system provides a steering angle feedback.

The four-wheel steering and driven configuration was introduced in order to provide a flexible platform for the research, but the improved mobility also provides a large quantity of more practical benefits. For instance, the 4WSD allows parallel displacement of the vehicle during turns by decoupling adjustments in position from adjustments in orientation. It also allows both the front and rear of the vehicle to follow a specific path precisely and may maintain a fixed orientation relative to the crop rows. In adittion to this, the ability of steering on all wheels also minimizes side slip of the wheels resulting in reduced wear on the vehicle and less damage to the field (Bak et al., 2004).

At this moment we are embedding in AgriBOT its onboard computers and navigation sensors. When it comes to embedded sensors, AgriBOT has 2 SICK LMS 291-S05 laser sensors fixed on its front part, pointing down at -30° (this allow us to obtain a 3D map of the area in front of AgriBOT, and verify its transversality), one IMU model DMS-

SGP02, one GPS RTK, and two CCD Cameras model GC1600CH (1600x1200, GigE, 1/1.8" CCD, color, 25 fps) as one may visualize in Fig. 2.



(a) (b)

Figure 2: Views of the AgriBOT. In (a) its top view and the detail of the GPS and IMU sensor positions. In (b), its front view and the detail of the 2 SICK LMS 291-S05 laser sensor positions.

In order to process the sensor data we are using 2 computers (processor i7-860 Quad Core 64bits, 2.8GHz, 4GB RAM, 1TB HD, Ethernet, Firewire, with anti-vibration system). One computer is dedicated to process the artificial vision data, and the other one processes the IMU, GPS, and laser sensor data and runs the navigation and obstacle avoidance routines. Here it is important to highlight that we needed to duplicate the laser/camera quantity for the navigation purposes because the AgriBOT mobile robot is suppose to move in the middle of the plantation (orange, sugar-cane, etc.). So, it is necessary to monitor the area in front of each wheel separately. The navigation software was entirely developed using C++ in Linux platform.

3. PATH PLANNING ALGORITHMS

3.1. The D* Algorithm

The algorithm D* is similar to the A* algorithm in several aspects. In both cases, an open list that contains the candidate nodes to be part of the trajectory is used and the both apply heuristic functions in order to define the minimum path. Although the initial trajectory calculation is carried out similarly for both algorithms, the D* algorithm considers that the map can change as the robot moves and it has internal routines that allow modification of the path if necessary.

Likewise the algorithm A*, the D* algorithm nodes are divided into three classes:

- New node, if it has never entered the *open list*;
- Open node, if it is in the *open list*;
- Closed node, if it is in the *closed list*.

However, in the D* algorithm a node can leave the *closed list* and return to the *open list* when the map is modified. This happens because a previously invalid path can become valid (collision-free) due to the map change. In addition to this, the D* algorithm, the father of a node X is defined as the node that succeeds X on the optimized trajectory that passes through X. Here again there is a difference between both algorithms. In the A* algorithm, the father precedes the node X. This change occurs because the rules used in the A* algorithm are not the same ones aplied in the D* algorithm. In the D* algorithm, the heuristic function is calculated based on the cost of moving the robot to the destination node, i.e., the heuristic of any node adjacent to X destination will be the cost of moving the node to the destination. So, the heuristic function value (h) of any node X that has Y as a parent will become the cost of moving from X node to the destination node added by the cost of moving from X node to Y node, and so on, as presented in Fig. 3. Thus, one may observe that, in contrast to A* algorithm, the D* algorithm does not use the f and g parameters to calculate the heuristic function. These parameters applied in the A* algorithm force the algorithm to find the optimal path giving preference to the destination node direction. As the D* algorithm renders the states from the target position without any preferable direction, it tends to be computationally less efficient. However, in order to improve the route, when the D* algorithm is applied on the map, it is necessary to use these parameters in the heuristic function, as it will be presented later on.

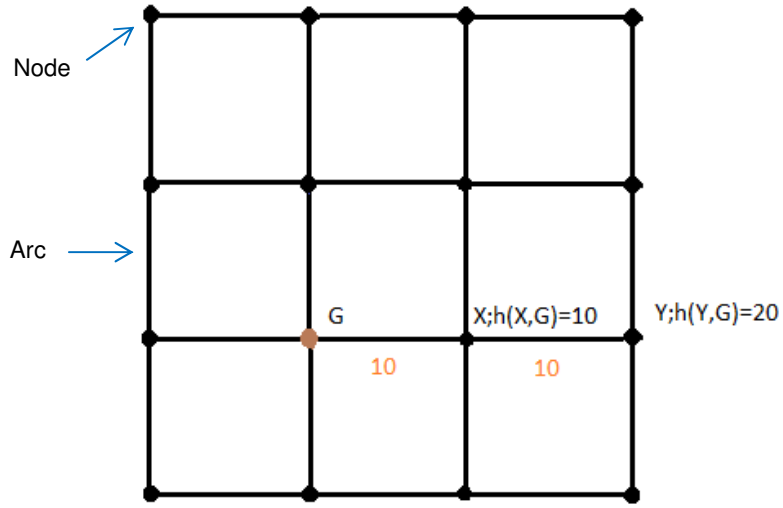


Figure 3: Calculation of the heuristic function in D* algorithm. The orange values bellow the arcs between nodes represent the costs of moving the robot between two nodes (e.g.: between nodes X and $G \rightarrow h(X, G) = 10$; and between nodes Y and $G \rightarrow h(Y, G) = 20$). Node G, represented by a brown punct, is the destination node.

In addition to using a diffrent heuristic function, the algorithm D* also uses a function called key function (k). Basically, it can be understood as follows. For each state X of the *open list*, and setting the letter G to the target node, the function $k(X, G)$ is defined as the smallest value of function $h(X, G)$ provided when X was added to the *open list* and before the map has been modified.

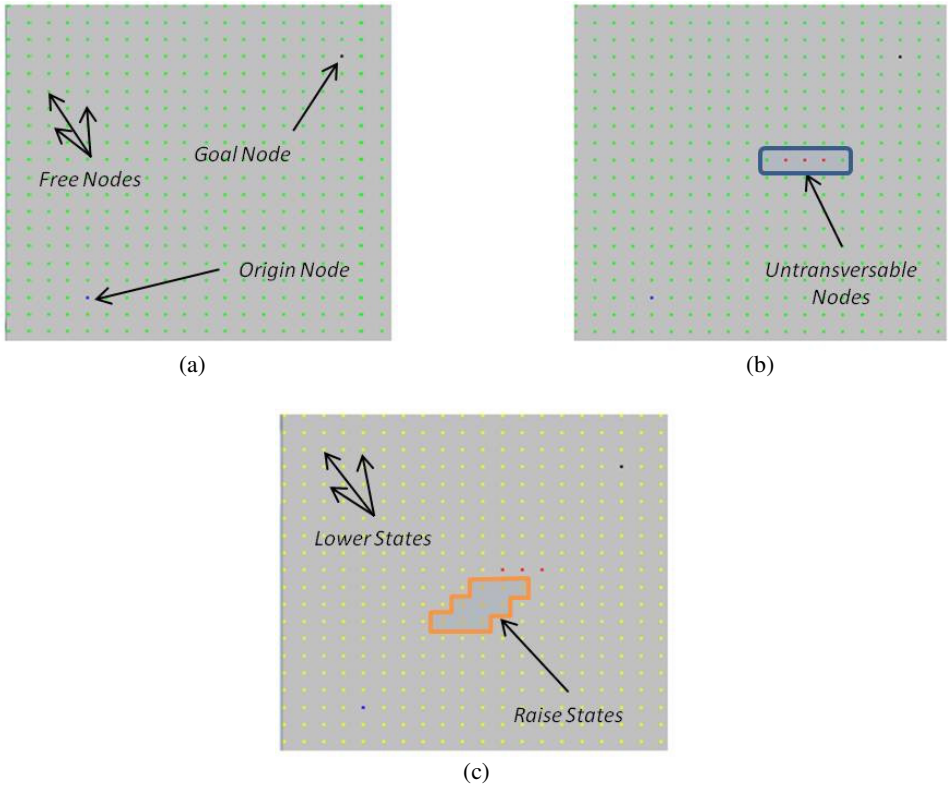


Figure 4: Example of a wave raise propagation. In (a): the initial map provided to the mobile robot (the green dots represent the free nodes, the red ones, the untransversable nodes, the blue dot, the origin node, and the black dot, the goal node). In (b): the current map of the environment where the untransversable nodes are represented by red dots. In (c): a detail of the wave raise propagation, where lower states are represented by yellow dots and the raise states are represented by orange dots.

In the D* algorithm, the states of the *open list* are classified into two types: raise and lower nodes. Every time the cost of crossing a X node increases, this node is classified as a raise node, and this incensement is propagated to all Y nodes that have X as father, and then to to all Z nodes that have Y as father, and so on. The logic behind this is that, as the cost of crossing X has changed, the costs of all paths that pass through X must be also modified.

Hence the need of using the heuristics described above. So, the cost is spread by increasing the cost of the heuristic function, i.e., Y node will have its heuristics modified to a new value $h(X, G) + c(X, Y)$ (that was corrected and is now higher). If a state X has the same value of h after the map was modified, it is classified as a lower node. Thus, when $k(X, G) = h(X, G)$, X is the lower node. If $k(X, G) < h(X, G)$, X is classified as a raise node. The basic idea behind this classification is that states raise the values of R before the map has been changed, i.e., $k(R, G)$ defines a lower limit on costs that can be obtained in the trajectories formed by lower states.

Therefore, if the costs of the paths formed by lower states in the *open list* exceed the costs of previous paths formed by the raise states, then it is necessary to check again the raise states in order to obtain the least cost route. Concluding, one may observe that the D* algorithm deals with map changes based on the propagation of two waves. Firstly, it spreads a wave of raise states, whose function is to propagate the rising cost of the node to the trajectories that pass through it. Then, it spreads a wave of lower states, which aims to determine if the routes had increased their costs and if they can still be optimized. Figure 4 illustrates the wave propagation of a raise state. It is possible to observe in the figure that the wave propagated in the trajectories passing through the obstacles before the map has been modified.

3.2. The Focussed D* Algorithm

The Focussed D* algorithm was created in order to deal with possible errors of the environment map (incomplete maps) that is provided to the mobile robot. Basically, it has a routine that updates the route every time that a change in the map is detected. As the Focussed D* algorithm produces results similar to the D * algorithm, one may consider that it is an improvement of the D* algorithm. So, given a map divided into several nodes (each one classified as transversable/free or untransversable), and given the mobile robot start position and goal position, as shown in Fig. 5, the Focussed D* algorithm selects the sequence of nodes that results on the lowest cost route among all possible routes even if the environment changes or the initial map is incomplete or inaccurate.

The logical procedures of the Focussed D* and D *algorithms are very similar. In both cases, the algorithms have an *open list* which contains the candidate states to join the series of states that form the final path. Moreover, both algorithms use the mechanism of raise/lower wave propagation to correct the path when an error in the map is detected. Another similarity is the use of a heuristic $h(X, G)$, where X is any state and G is the goal, and the function $k(X, G)$. The difference between them essentially lies in how they deal with the cost of any change of the map state. When it comes to the D* algorithm, as previously described, the propagation cost is obtained without considering any preferable direction. On the other hand, the Focussed D* algorithm uses a heuristic that focus the cost spread towards the mobile robot goal position, due to this, it becomes computationally more efficient.

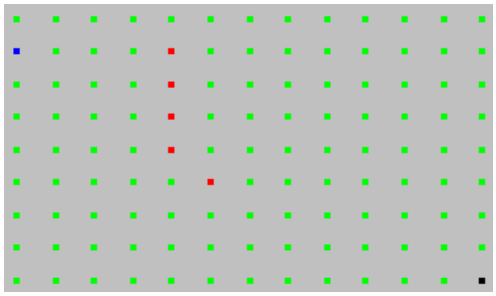


Figure 5: Environment represented by a topological map. The green dots represent the transversable/free nodes, the red ones, the untransversable nodes, the blue dot, the origin node, and the black dot, the goal node.

Let us take Figure 5 to illustrate this behavior. In this figure, the mobile robot is placed in the blue node, and we want to move it to its goal position (the black node). During the path, it detects obstacles and updates the initial map changing the classification of some nodes from free to untransversable nodes (the red nodes). As, in this case, the nodes that lie behind the mobile robot probably will not be part of the final trajectory, so there is no reason for considering them in the search for the minimum cost path. Let us consider the focused heuristics $g(X, R)$, where X is any state and R the current position of the robot. This heuristics represents the cost estimate for the mobile robot to move from R to X. So, we define a new function, the cost estimate, as $f(X, R) = g(X, R) + h(G, R)$. In this case, the function $f(X, R)$ represents the estimated cost of moving the mobile robot from its current position to the desired goal position passing through the state X.

In the D* algorithm, the previous values of $h(\ )$ determines a lower limit on the values of $h(\ )$ that could be obtained at lower states, so if the same function $g(\ )$ is used for both types of state, previous values of $f(\ )$ can also be

used in setting the lower limit values of $f(\ )$ that can be obtained at lower states. So, in the Focussed D* algorithm, the *open list* is sorted by the values of $f(X, R)$. If the value of $f(\ )$ in the two states are equal, the value of both k and the state are compared with the lowest value of k.

Similarly to the D* algorithm, here an optimal path is considered found when the lowest value of the *open list* is equal to or greater than the cost of the mobile robot path. The procedure used to focus the cost spread can be explained considering the example presented in Fig. 6. Firstly we assume that a mobile robot uses a Focussed D* algorithm and it finds an error in its initial map. So, it updates its route and moves to a new position. Then, another error is detected in its new position, but the states of the *open list* are focused on the previous position, with values of $f(\ )$ and $g(\ )$ that are incorrect. One way to solve this problem is to recalculate the values of $f(\ )$ and $g(\ )$ everytime that the mobile robot moves and an error is found. However this solution is an inefficient approach, due to the computational cost related. In order to solve this issue, Stenz (1994) proposed the following procedure. We add a state X to the *open list* when the mobile robot is in the position R_0 (Fig. 6). If the mobile robot moves to the position R_1, we define a lower limit for the value of $f(X, R_1)$ given by $f_l(X, R_1) = f(X, R_1) - g(R_1, R_0)$. $f_l(X, R_1)$ is a lower limit because it assumes that the mobile robot moved toward the state X. If X is placed in the *open list* considering the value of $f_l(X, R_1)$, X will be evaluated at the appropriate values of $f(X, R_1)$, $g(X, R_1)$ will be upgraded, and X will be placed back in *open list* with the correct value of $f(X, R_1)$.

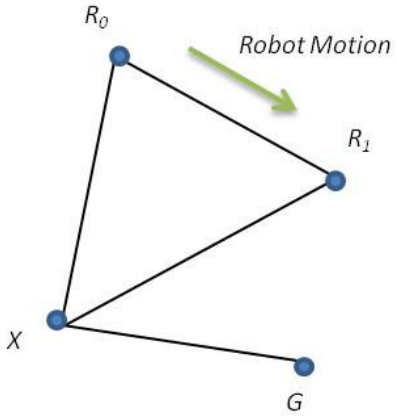
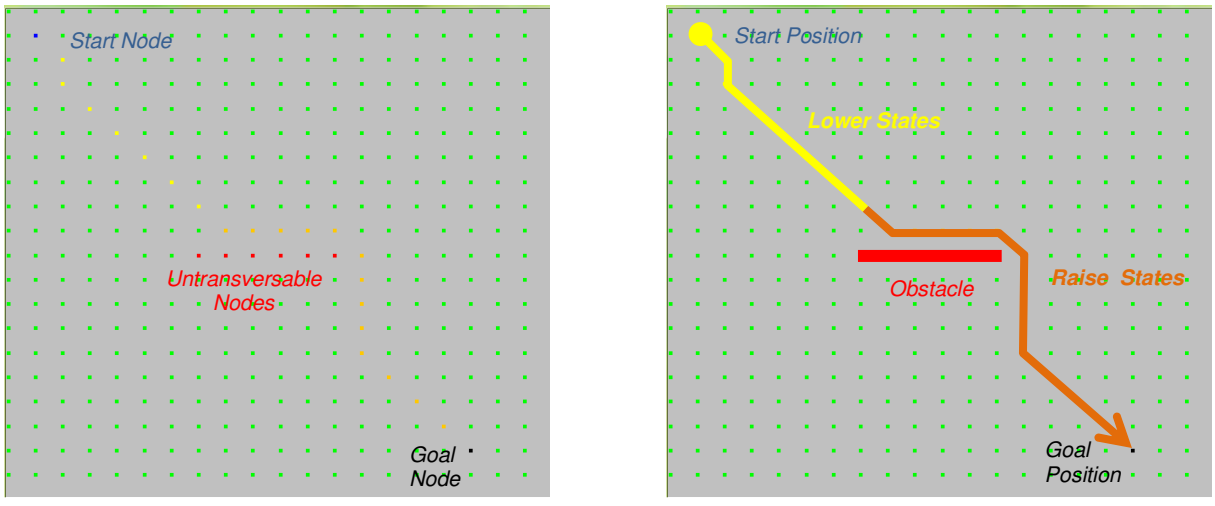


Figure 6: Illustration of the solution proposed by Stenz (1994).

Figure 7 illustrates the result that is obtained by applying the Focussed D* algorithm and the procedure proposed by Stenz (1994) to correct the values of $f(\ )$ and $g(\ )$. For this environment, the robot had an incomplete map (in this case, an obstacle that was not in the map). During the path, the mobile robot detected the obstacle and upgraded the map and the *open list*. In the end, a new path is obtained.



(a) (b)

Figure 7: Result obtained by applying the Focussed D* algorithm and the procedure proposed by Stenz (1994). In the beginning of the path the mobile robot does not know that there are obstacles in the environment. In (a): the green dots represent the free nodes, the red ones, the untransversable nodes, the blue dot represent the origin node, and the black

dot, the goal node. Lower states are represented by yellow dots and the raise states are represented by orange dots. In (b): the initial path is represented by a yellow line and the orange line is the upgraded path.

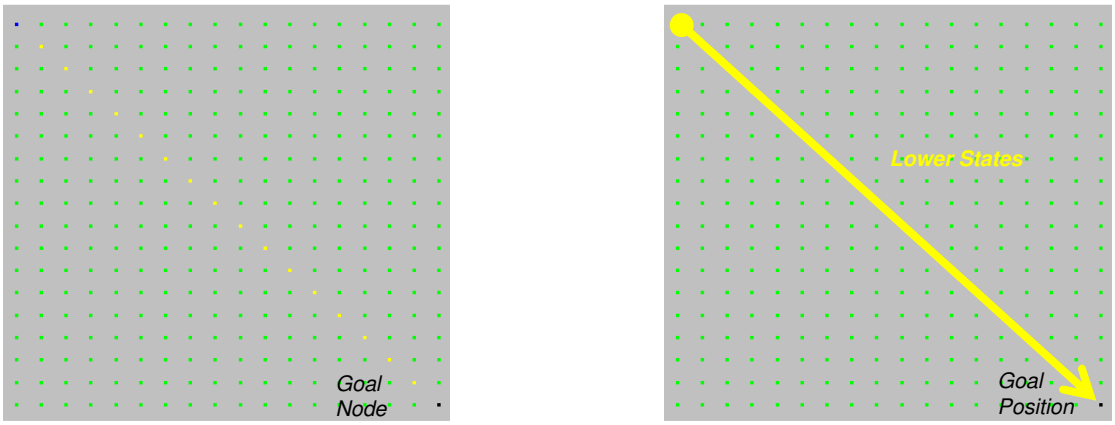
4. RESULTS

The next figures (Fig. 8 to 10) show the results obtained applying both algorithms, D* and Focussed D* in order to obtain the minimum path for a mobile robot between the start position and the goal position for three different environment. The complexity of the environment was increased from Fig. 8 to 10. All simulations were carried out in the same computer and under the same conditions. One may notice that the paths obtained were equals in all cases. Due to this, we decided to compare the processing time needed to generate the paths using both algorithms. So, Tab. 1 shows the results in terms of processing time.

The results presented in Fig. 8 to 10 show that when the environment has very large and large obstacles (like in Fig. 9) the D* algorithm is more computationally efficient. On the other hand, when the environment has several medium and small obstacles (like in Fig. 10), the Focussed D* algorithm presents better results. This behavior can be understood taking into account that the Focussed D* algorithm limits the states expansion area on the map when an unmapped obstacle is found. Due to this, it deals with medium and small obstacles more quickly. But, when it comes to very large and large obstacles, the D* algorithm is more efficient because the Focussed D* algorithm needs to increase its search area radius several times, therefore this procedure becomes very time consuming. In Fig. 8, the environment has no obstacle and the Focussed D* algorithm was more efficient because it directs the trajectory towards the goal position.

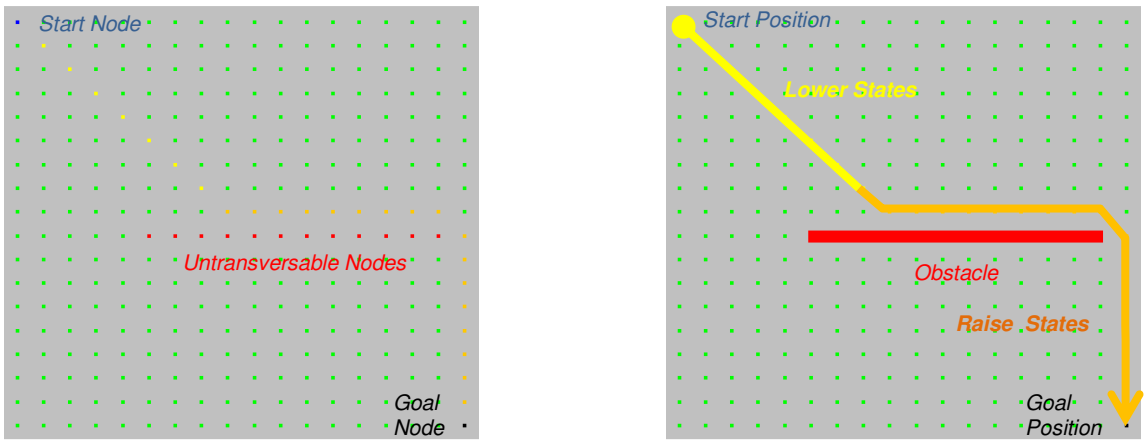
Table 1 - Results of the D* and Focussed D* Algorithms.

Approach	Processing Time Needed		
	Environment # 1 (Fig. 7)	Environment # 2 (Fig. 8)	Environment # 3 (Fig. 9)
D* Algorithm	47 ms	78 ms	156 ms
Focussed D* Algorithm	16 ms	140 ms	78 ms



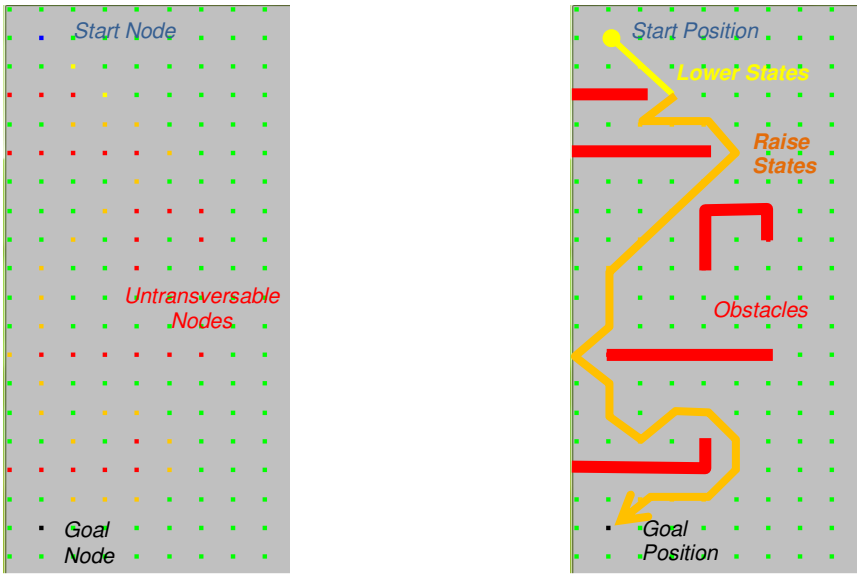
(a) (b)

Figure 8: Result obtained by applying the D* and Focussed D* algorithms. In this case, the map was complete (i.e.: there was no unmapped obstacles in the environment). In (a): the green dots represent the free nodes, the blue dot represent the origin node, and the black dot, the goal node. Lower states are represented by yellow dots. In (b): the path is represented by a yellow line.



(a) (b)

Figure 9 - Result obtained by applying the D* and Focussed D* algorithms. In this case, there was an unmapped obstacle in the environment. In (a): the green dots represent the free nodes, the red ones, the untransversable nodes, the blue dot represent the origin node, and the black dot, the goal node. Lower states are represented by yellow dots and the raise states are represented by orange dots. In (b): the initial path is represented by a yellow line and the orange line is the upgraded path.



(a) (b)

Figure 10: Result obtained by applying the D* and Focussed D* algorithms. In this case, seven unmapped obstacles were in the environment. In (a): the green dots represent the free nodes, the red ones, the untransversable nodes, the blue dot represent the origin node, and the black dot, the goal node. Lower states are represented by yellow dots and the raise states are represented by orange dots. In (b): the initial path is represented by a yellow line and the orange line is the upgraded path.

5. CONCLUSIONS

In this paper we initially introduced the use of mobile robotics systems in agricultural applications. In addition to this, we presented briefly the AgriBOT project and the AgriBOT mobile robot. Next, we described the D* and the Focussed D* algorithms, and highlighted their similarities and differences. Then, in the results section we presented the simulation results obtained and we compared both algorithms based on the processing time needed for the same environment and initial conditions. As expected, the Focussed D* algorithm was more efficient in environments which maps were incomplete or inaccurate. But, we noticed that when the unmapped obstacles can be considered very large or large (taking into account the environment dimensions) the D* algorithm produced better results.

Regarding that an agricultural field may be considered a quasi static environment, it is feasible to expect that changes in the environment may happen. In such cases, the initial paths generated for the AgriBOT may need to be updated as new environment data is received that are in conflict with the original maps. So, it is necessary to upgrade the path as fast as possible. As a consequence of these reasons, we conclude that for our application the best approach is the Focussed D* algorithm.

6. ACKNOWLEDGEMENTS

The authors would like to acknowledge FAPESP (Process No. 2009/04787-0), CNPq (Process No. 180833/2010-3), FINEP, Embrapa, Jacto, and Colciencias for their financial support for this research.

7. REFERENCES

Abrahão, G. Q. S., Megda, P.T., Becker, M., 2010, "O Uso de Algoritmos A* e Field A* em Robô Agrícola", CONEM 2010, Campina Grande, Brasil.

Bak, T and Jakobsen, H, "Agricultural Robotic Platform with Four Wheel Steering for Weed Detection" Biosystems Engineering, Volume 87, Issue 2, February 2004, Pages 125-136.

Ferguson, D and Stentz, A, 2005-a, "Field D*: An Interpolation-based Path Planner and Replanner", Proceedings of the International Symposium on Robotics Research (ISRR), 2005.

Ferguson, D. and Stentz, A., 2005-b, "The Delayed D* Algorithm for Efficient Path Replanning", Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005. pp. 2045 – 2050.

UNIBOTS, 2011, http://www.unibots.com/Agricultural_Robotics_Portal.htm, accesed in Jan. 2011.

Nilsson, N, 1980, "Principles of Artificial Intelligence", Tioga Publishing Company, pp. 72-88, 1980.

Stentz, A, 1994, "The D* Algorithm for Real-Time Planning of Optimal Traverses", PhD Thesis - Carnegie Mellon University.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.