Robotica: (2020) volume 38, pp. 684–698. © Cambridge University Press 2019
doi:10.1017/S0263574719000961

Path Planning Aware of Robot's Center of Mass for Steep Slope Vineyards

Luís Santos†*, Filipe Santos†, Jorge Mendes†‡, Pedro Costa†¶, José Lima†§, Ricardo Reis† and Pranjali Shinde†

†*CRIIS - Centre for Robotics in Industry and Intelligent Systems, INESC-TEC, Porto, Portugal*
E-mails: fbnsantos@inesctec.pt, ricardo.g.reis@inesctec.pt, pranjali.shinde@inesctec.pt
‡*UTAD, Vila Real, Portugal. E-mail: jorge.m.mendes@inesctec.pt*
¶*FEUP, Porto, Portugal. E-mail: pedrogc@fe.up.pt*
§*CeDRI and IPB, Bragança, Portugal. E-mail: jllima@ipb.pt*

(Accepted May 25, 2019. First published online: July 18, 2019)

SUMMARY
Steep slope vineyards are a complex scenario for the development of ground robots. Planning a safe robot trajectory is one of the biggest challenges in this scenario, characterized by irregular surfaces and strong slopes (more than 35°). Moving the robot through a pile of stones, spots with high slope or/and with wrong robot yaw may result in an abrupt fall of the robot, damaging the equipment and centenary vines, and sometimes imposing injuries to humans. This paper presents a novel approach for path planning aware of center of mass of the robot for application in sloppy terrains. Agricultural robotic path planning (AgRobPP) is a framework that considers the A* algorithm by expanding inner functions to deal with three main inputs: multi-layer occupation grid map, altitude map and robot's center of mass. This multi-layer grid map is updated by obstacles taking into account the terrain slope and maximum robot posture. AgRobPP is also extended with algorithms for local trajectory re-planning during the execution of a trajectory that is blocked by the presence of an obstacle, always assuring the safety of the re-planned path. AgRobPP has a novel PointCloud translator algorithm called PointCloud to grid map and digital elevation model (PC2GD), which extracts the occupation grid map and digital elevation model from a PointCloud. This can be used in AgRobPP core algorithms and farm management intelligent systems as well. AgRobPP algorithms demonstrate a great performance with the real data acquired from AgRob V16, a robotic platform developed for autonomous navigation in steep slope vineyards.

KEYWORDS: Agricultural robots; Path planning; Center of mass; Vineyard.

1. Introduction

The steep slope vineyards placed in the Douro Demarcated region (Portugal), UNESCO Heritage place, present unique characteristics, Fig. 1, which includes a number of robotic challenges to be overcome in order to reach a full autonomous navigation system. Due to unique topographic and soil profiles, these challenges include research in robotics areas, such as visual perception, localization, environmental modeling, control or decision-making. Developing an autonomous system requires to consider the scenario characterization, platform constraints and the types of sensors used for collecting the scenario information.[1] To navigate safely on the farm, the path planning algorithm requires an accurate localization and a good map of the vineyard. Global Navigation Satellite Systems (GNSS) are not always available (signals are blocked by the hills) and the dead-reckoning sensors accuracy is

* Corresponding author. E-mail: luis.c.santos@inesctec.pt

Fig. 1. Typical Douro's steep slope vineyard.

affected by the harsh terrain conditions; so for that reason, in the past we have proposed a VineSLAM algorithm.[2]

This work aims to research and develop a path planning framework, able to make the robot travel from one location to another avoiding obstacles and dangerous slopes that would present risks to the robotic platform, centenary vine trees or even other live beings. To accomplish this goal, this paper presents an extended A* algorithm that takes into account the orientation and center of mass of the robotic platform for safe navigation in a steep slope vineyard avoiding danger postures for the robot. To reach a safe path, the extended A* algorithm needs two maps, occupation grid map and digital elevation model (DEM). This work presents an algorithm to extract these two maps from a 3D PointCloud obtained by any agricultural robot.

In this paper, Section 2 presents the related work on the path planning. Section 3 presents the proposed extended A* algorithm, called agricultural robotic path planning (AgRobPP), that is aware of center of mass of the robot and terrain slope. Section 4 presents how to extract the grid map and DEM from a PointCloud. Section 5 presents tests and results. Section 6 presents conclusions of the paper.

2. Related Work

Path planning is the capability to find the best path between a start point and an end point. However, finding a path is not enough; it is necessary to choose the best path suitable for the required task based on several constraints. Several approaches can be used to find the optimal path between two points, such as potential field planners, tree-based planners rapidly exploring random tree (RRT), coverage path planners and search algorithms like Dijkstra and A* which are widely used for path planning tasks.[3–5]

In potential field planners, the robot behaves like a particle immerse in a potential field, where the destiny represents an attraction potential and the obstacles represent repulsive potentials. A common problem with these planners is the existence of local minimums, which emerges in situations where the repulsive potential is greater than the attraction potential. This situation will stop the robot from reaching the destiny.[4] There are some works with modified potential field planners to work in dynamic environments and solve minimum local problems. For example, the robot can wait for a change in the environment or do slight random movements to eliminate the minimum local. Other alternative is to contour the obstacle walls until the problem disappears.[6]

In RRT algorithms, the path is randomly explored, planning the path between two points considering the robot's dynamic. It is adequate for non-holonomic robots. These planners are usually simple algorithms easy to understand; however, they are not optimal and generate paths with many abrupt curves, something that is not appropriated for robot navigation, particularly in outdoor environments.[3] There are variations to decrease the direction changes and improve the quality of the planner. Rodriguez et al.[3] propose an RRT algorithm with obstacle information that generates smoother paths and Karaman et al.[7] present RRT*, a modified tree-based algorithm that converges to a near optimal method. Song et al.[8] have presented an RRT*-based algorithm that considers both environment-level and vehicle-level constraints, called triangle-curvature RRT (TC-RRT). It has been found that for more complex environments TC-RRT is faster and the path generated is shorter in comparison to a simple RRT. Auat Cheein et al.[9] propose an RRT-based algorithm that takes into account the various obstacles of an agricultural environment such as terrain deformability and slopes. It aims to plan a path from a point of harvest to another point safely while optimizing the losses and the efforts of

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

the platform. The paths are generated using the RRT considering the variations of the terrain, which avoided the difficult paths to cross and thus reducing the total effort and time needed for the completion. Like other RRT-based algorithms this algorithm does not ensure to find the optimal path. Wang et al.[10] presented a system for indoor navigation which avoids the stairs and uses the inclined planes and this system is applied to a wheel chair. This work modifies the RRT to generate a smooth path and obstacle-free dynamic deviation. The environment description (map) is obtained using the robotic odometry, 2D laser and RGB-D-based sensors to generate a 3D map with OctoMap[1] that feeds the RRT-based algorithm.

Coverage path planning (CPP) is another type of path planning algorithms found in literature. CPP is useful for applications where the robot needs to visit every points of a certain area at least once.[11] This is an useful technique for precision agricultural usually used with unmanned aerial vehicles (UAVs), not discarding ground robots applications. Valente et al.[12] propose a near optimal CPP for a low-cost UAV to capture high-resolution images of irregular-shaped fields. Jin and Tang[13] present a CPP system on 3D terrain for arable farming. This includes four main tasks: terrain modeling, coverage cost analysis, terrain decomposition and optimization of the search algorithm.

Search algorithms like A* or Dijkstra can be used in path planning problems when environment representation (map) uses grid- or graphs-based maps. The grid-based maps are usually divided in cells (squares), where each cell contains information about its availability (existence of obstacle or free space). Considering these maps, algorithms like A* or Dijkstra can search one path between two points. A* is a variant of Dijkstra improved to always find the best path between two points, which means that A* is an optimal algorithm. However, A* has an increased computational complexity when compared to the previous alternatives. There are many variations of the original A* algorithm to improve certain characteristics, such as processing time,[14] non-stationary environments[15,16] and awareness of other parameters like the robot's orientation.[5] Santos et al.[17] present a A*-based algorithm capable of planning an off-line path for a docking station located in a steep slope vineyard. In this work, a set of visual tags were used to generate the trajectory to be described by the robot to the docking station. The best feasible path to the nearest docking station was calculated using the A* algorithm considering the energy cost of the robot to navigate from its localization to the docking station. Schirner et al.[18] studied the navigation of a lawnmower so that it could avoid zones of potential loss of localization. For this purpose, a robotic platform equipped with low-computational power and sensing used mapping and localization (SLAM – simultaneous localization and mapping) techniques to navigate and describe the surrounding environment. The authors chose a Dijkstra algorithm with an evaluation function based on the distance and the uncertain in order to privilege nodes that belong to a shorter path. Tian[19] presents the construction of a map and a global path planning algorithm in a 2D plane. Environmental data were acquired using a 3D LIDAR (VLP-16), where the 3D information is projected in a 2D plane, using a grid map for the spatial representation, and a D* algorithm to the search for the path. D* algorithm is a modification of A* ready to consider the robot's dynamic.

It is possible to consider strategies like 3D mapping because it allows to obtain a complete representation of the environment. Stoyanov et al.[20] propose an approach using a complete 3D representation: three-dimensional normal distributions transform (3D-NDT). This type of spatial representation presents a compact and expressive description of the environment which can be used for point recording and mapping. An adaptation of the wavefront algorithm was used to work with representations of 3D-NDT. In the study, the use of 3D-NDT was beneficial because it reduced the complexity of the planning algorithm, removing the need to design in 2D and the possibility of loss of diversity of paths.

In literature, there is no solution that fulfills the presented requirements for path planning to be aware of robot's center of mass and slope terrain, required for steep slope vineyards scenarios. It is required that the robot should be capable of navigating safely taking into account all terrain irregularities, obstacles and potential signal localization failures.

3. AgRob Path Planning (AgRobPP)

To choose the adequate approach for path planning, three parameters have been considered: the desired optimization (time, distance, etc.), the computational complexity and the method efficacy.

[1]OctoMap – http://wiki.ros.org/octomap.

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

Also, the environment plays a significant role in adopting a good strategy for path planning. From literature, it is possible to state that A* algorithm can be extended to optimize specific parameters such as time, energy or other parameters, such as unsure maximum limits to robot posture. A* may have an elevated computation complexity, but is still a good solution for path planning tasks considering that it is the only optimal algorithm, which assures that always finds the best path between two points according to the user requirements.

In our previous work, Fernandes et al.[5] presented an extended A* algorithm that constraints the found path to a maximum turn rate and optimizes the safety regions by considering the robot orientation (yaw), which is useful to navigate on the vineyard where the robot must move through confined space. This work extends A* inner functions for path planning to be aware of robot's center of mass and terrain slope. To reach this goal, this extension considers two maps, occupation grid map and a digital elevation map, which is obtained from 3D PointCloud of the vineyard.

AgRobPP is an algorithmic approach which contains the developed solution for safe path planning in steep slope vineyards. It is composed of:

- update A* inner functions for the path planning to be aware of robot orientation and center of mass;
- a method for safe local re-planning; and
- PC2GD.

Usually, a standard A* algorithm, to generate the best path between two points, has in the input: an occupation grid map and the robot's position (x, y) in the world. The extended A* algorithm presented in this document, AgRobPP, takes into account not only the position of the robot but also the robot orientation and center of mass. When this information is combined with an extra input, a DEM, it is possible to generate a feasible and safe path for the robot. This enables the autonomous navigation along uneven and high inclination terrains such as steep slope vineyards.

AgRoBPP extends Fernandes et al.[5] work which proposes an extended A* algorithm that limits the generated path with a maximum robot turn rate. This goal is reached by creating multiple grid-map layers which discretizes the orientation into spaces of 22.5°; this defines the A* jump cells during the path search. The robot orientation discretization is made by the number of neighbors of a cell (square) equally radially spaced; this gives a number of $2^{(n+2)}$ neighbors, where n is the radius search space of a discretized circle. So, by dividing the circle into equal parts, it is possible to obtain a discretization of 45° ($n = 1$), 22.5° ($n = 2$), 12.25° ($n = 3$), 6.125° ($n = 4$), so on. The chosen value is 22.5°, because a higher value would generate trajectories with abrupt orientation changes whereas a lower value would increase the search space of the A* algorithm which would increase the distance between waypoints (more computational requirements), decreasing the distance resolution in the generated path. The Subsection 3.1 presents a brief explanation of this algorithm and Section 3.2 presents A* extension for path planning aware of robot's center of mass and terrain slope.

3.1. Expansion of A algorithm to be aware of orientation*

A typical A* algorithm contains two lists: the open list that contains the nodes candidates for exploration and the closed list that contains the explored nodes (cells). The nodes in these lists store the "parent" node, which is the node used to optimally reach them. A typical pseudo code from A* algorithm is described in Algorithm 1, with the following variables[16]:

- $c(n1, n2)$ – cost from going from node $n1$ to node $n2$.
- $f(n) = g(n) + h(n)$ – estimation of the lowest cost of going from the origin to the target passing through node n.
- $g(n)$ – cost from the origin to node n.
- $h(n)$ – an heuristic to estimate the cost of the path from the node n to the target node
- $Q(n)$ – set of neighbors of node n.

A traditional A* algorithm based on grid map searches for a path using a simple 2D map. This map is composed of a grid with cells (nodes) that contain information about its availability, that is, it is a free cell or an obstacle. During the search the algorithm analyzes the neighbors of a cell ($Q(n)$) in order to calculate which one will be present in the generated path. Considering a map with fixed

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

Algorithm 1 A* algorithm[16]

1: Add origin node to O (Open list)
2: **Repeat**
3: Choose *nbest* (best node) from O so that $f(nbest) \leq f(n)\ \forall\ n \in O$
4: Remove *nbest* from O and add it to C (Closed list)
5: **if** *nbest* **= target node then end**
6: For all $x \in Q(nbest)$ which are not in C do:
  if $x \notin O$ then
    Add node x to O
  else if $g(nbest) + c(nbest, x) < g(x)$ then
    Change parent of node x to *nbest*
7: **until** O is empty

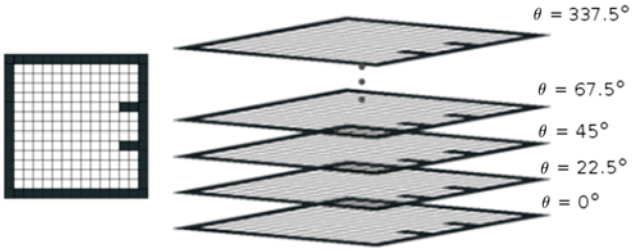

Fig. 2. Layered map representation.[5]

cells (squares), where every cell has the same size, there usually are eight neighbors to visit. Another characteristic usually present in this algorithm is the obstacle expansion. Although this is not part of the A* search algorithm, it is an almost mandatory operation to allow robot navigation. This consists in the expansion of the obstacles to the dimensions of the robot to assure its safety during navigation. Traditionally, for obstacle expansion the robot is approximated to a circle and all the obstacles are inflated with the radius of the circle. The innovation presented in this A* version introduced three big changes related to the topics discussed above:

- Occupation Grid Map: Transforms the original 2D grid map into a 16-layered map. Each layer represents a 2D grid map with the same obstacle information but different robot's orientation, spaced by 22.5° each, as represented in Fig. 2. Each layer will contain an occupation grid map and represent the robot's orientation.
- Neighbors: Limits the search to three or six neighbor cells (if the robot is ready to navigate on reverse), to avoid abrupt orientation changes. This will be reflected in point 6 of Algorithm 1.
- Obstacles Expansion: The usual obstacles expansion that occurs in the maps used by A* is done individually in each layer, giving different results for each orientation, which allows to make a better expansion for rectangular-shaped robots. This way the different layers have the same obstacle information, but different obstacle expansion.[5]

3.2. AgRobPP architecture with path planning aware of orientation and center of mass

The center of mass constraint (maximum robot pitch/roll) is reached, Fig. 3, by considering two key inputs: the Altitude Map/DEM and the robot's center of mass provided by its coordinates (x, y, z) related to the base referential robot.

The following steps demonstrate the generation of the safest path with the knowledge of the robot's center of mass:

1. Compute normal vectors to every single cell of the occupation grid map.
2. Check the imposed projection of center of mass in each cell taking into account the 16 layers of the map, that is, considering different robot's orientation.
3. Check the safety limit of the robot in each cell, blocking the dangerous cells.

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

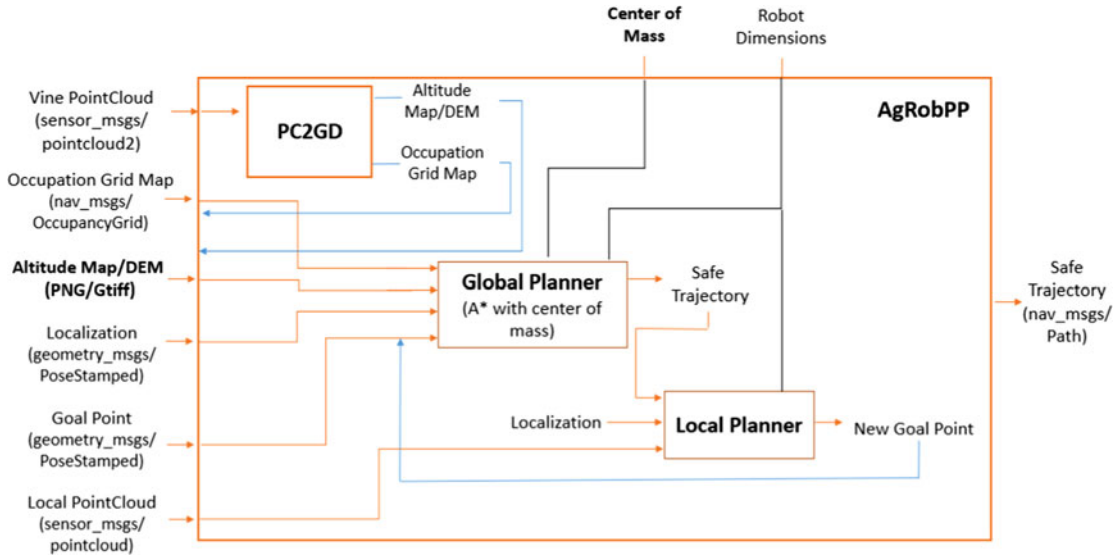

Fig. 3. AgRobPP diagram.

Normal vector estimation: The process of computing the normal vector of one cell is executed using information of at least two neighbor cells, usually chosen according to the robot's orientation. As example, the cell $P1\ (x, y, z_1)$ will be considered. For this cell the neighbors $P2\ (x-1, y+1, z_2)$ and $P3\ (x+1, y+1, z_3)$ are considered, but their disposition changes in case of unavailability (map borders) as depicted in Fig. 4. The normal vector $\overrightarrow{V_n}$ is calculated with the cross product expressed in Eq. (1):

$$\begin{gathered}\overrightarrow{V_1} = P2 - P1 \quad \overrightarrow{V_2} = P3 - P1 \\ \overrightarrow{V_n} = \overrightarrow{V_1} \times \overrightarrow{V_2}\end{gathered} \tag{1}$$

The values z_1, z_2, z_3 are directly read from the Altitude Map/DEM, a map similar to the occupation grid map with information about the elevation instead of the cell availability. This operation is applied to every single cell of the grid map. The neighbors position might change in case of unavailability, for example, in map borders.

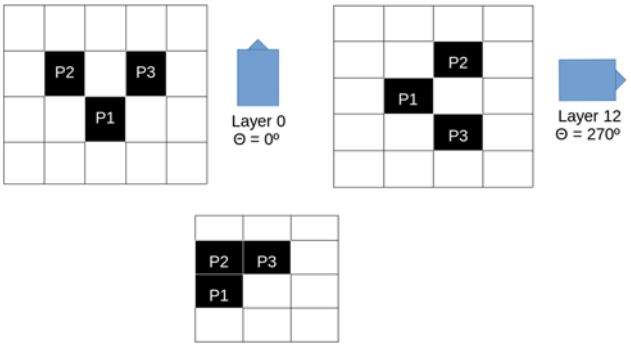

Fig. 4. Examples of a cell and two neighbors for normal vector estimation: left top: normal case for layer 0 of the map; right top: normal case for layer 12 of the map; bottom: special case in map border.

Safety check of center of mass: With normal vector $\overrightarrow{V_n} = (\overrightarrow{V_x}, \overrightarrow{V_y}, \overrightarrow{V_z})$ it is possible to express the robot's orientation in Euler Angles, where the *roll* and *pitch* are obtained according to Eq. (2) and the actual *yaw* from the robot's orientation and the trajectory robot yaw from the number of the map layer. It is important to make sure that all the normal vectors are calculated with the same point of view to guarantee consistent values in the estimated *roll* and *pitch*:

$$\begin{aligned}\text{roll} &= \text{atan2}(V_x, V_z)\ \text{rad} \\ \text{pitch} &= \text{atan2}(-V_y, V_z)\ \text{rad}\end{aligned} \tag{2}$$

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

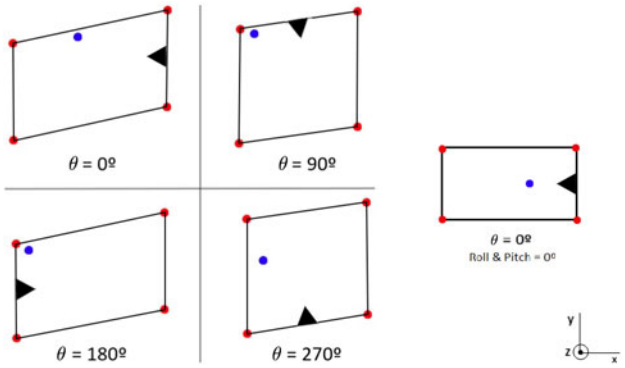

Fig. 5. Center of mass projection in inclined zone (left) and on a plain zone (right). Red – robot footprint; blue – center of mass projection; triangle – front of the robot; θ – robot's orientation (yaw).

Using the rotation matrix in Eq. (3) based on Euler Angles, where $\alpha = yaw$, $\beta = pitch$ and $\gamma = roll$, a rotation is applied to the center of mass represented in the base frame robot's referential, in order to obtain its projection in the horizontal plan of world frame. If this projection is not contained inside the polygon formed by the projected robot's shape, the robot will fall, and so this cell (x, y, θ) is marked as a danger spot (obstacle). Figure 5 shows an example of the center of mass projection with different orientation θ in a robot with dimensions 120 × 80 cm, a center of mass C_m (20, 0, 60) cm and a normal vector $\overrightarrow{V_n}$ (0.41; 0.41; 1.00) cm, that is a roll and pitch with absolute value of 24°. The red dots represent the wheels of the robot, the blue dots represent the center of mass projection and the black triangle signalizes the front of the robot. Figure 5 also includes a representation of the robot and its center of mass on a plain terrain on the right side:

$$R(\alpha, \beta, \gamma) = R_z(\alpha)R_y(\beta)R_y(\gamma) \tag{3}$$

$$R(\alpha, \beta, \gamma) = \begin{bmatrix} c(\alpha)c(\beta) & c(\alpha)s(\beta)s(\gamma) - s(\alpha)c(\gamma) & c(\alpha)s(\beta)c(\gamma) + s(\alpha)s(\gamma) \\ s(\alpha)c(\beta) & s(\alpha)s(\beta)s(\gamma) + c(\alpha)c(\gamma) & s(\alpha)s(\beta)c(\gamma) - c(\alpha)s(\gamma) \\ -s(\beta) & c(\beta)s(\gamma) & c(\beta)c(\gamma) \end{bmatrix}$$

Figure 5 illustrates the 2D-projected robot's center of mass for different robot orientations. This information will allow the path planner to avoid dangerous orientation in danger slope spots.

3.3. Local planning

Figure 3 shows that one of the AgRobPP inputs is a local PointCloud. This PointCloud is provided by the robot's laser scan that gives constant information about the environment near the robot, constantly checking for new obstacles (not present in the original occupation grid map). In case of a detection of a new obstacle, the robot stops its movement and uses AgrobPP to re-plans a new safety trajectory from the actual localization of the robot to a further waypoint in the original path. Usually it starts with a point that is at least two times the robot length from actual robot position. If it is not possible to generate a path, considering this waypoint, the algorithm will retry to generate a new path to the next waypoint. This runs iteratively until a new path is found, or until the final waypoint is reached (usually this means that is not possible to generate a path with the new obstacle). This re-planning takes into account the information of the new obstacle, which is reflected into the layered obstacle expansion. Figure 6 represents an example of a local path change, where the new obstacle is represented with the red color. The blue color illustrates the original path being followed by the robot (represented by it's base referential) and the green color represents the new safety re-planned path. In this test, the processing time of the entire operation was approximately 1.2 s (Intel core Core i5 3.2 GHz) which is acceptable for agricultural robots. The mandatory obstacle expansion execution before the generation of the new safest path for the robot (aware of the center of mass) resulted in the increased process time. To test and validate this feature, a model of a differential robot was simulated using STAGE simulator[2] with the simulated vineyard map.

[2]Stage – http://wiki.ros.org/stage.

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

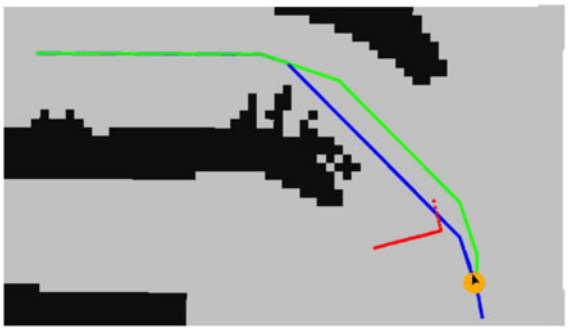

Fig. 6. Local planning example. Blue – original path; red – new obstacle; green – new path; yellow – simulated robot.

4. PointCloud to Grid Map and DEM

Considering the robotic localization approach presented by Santos et al., (VineSLAM[2]) and considering the vineyard natural feature detector (ViTruDe[21]), it was possible to extract an occupation grid map with "PC2GD" approach. Joining on AgRob V16 (described in Section 5.1) the VineSLAM and the OctoMap ROS package are possible to obtain a detailed PointCloud that can be used by PC2GD to extract a 2D occupation grid map and a DEM.

After the PointCloud extraction, the next step is to segment that data, attributing a classification to every single point of the PointCloud. Such segmentation will be useful to get information about the availability of a specific zone, which will allow to construct the 2D grid map. The DEM is obtained using the information about altitude from the PointCloud points.

PC2GD algorithm is divided into five steps: PointCloud extraction, PointCloud segmentation, generation of 2D occupation grid map, smoothing of 2D map and creation of DEM.

PointCloud extraction: It is necessary for an accurate localization system to obtain an accurate 3D representation of the environment. VineSLAM[2] approach is considered to perform the robot localization in Douro Steep Slope Vineyards. In contrast, the vineyard PointCloud is obtained using Octomap ROS package and considering a 2D vertical LIDAR. The next step is to annotate and segment the obtained PointCloud where every single point will be identified as floor or vegetation.

PointCloud segmentation: In this agricultural context, the PointCloud is segmented into two classes: floor and vegetation or wall (not floor). Segmentation is performed in this phase by collecting the terrain inclination information; if a certain point has what is considered to be an exaggerated slope, it will be classified as vertical vegetation or wall. To get the information about inclination, a normal vector is extracted in each point. Using a KD-tree method in each point the nearest neighbors are searched and the information is extracted about the altitude deviation between neighbor points. The PointCloud Library[3] offers an optimized and fast resource to perform this operation, computing all the normal vectors of a cloud with 150,000 points in less than 0.5 s.

Generate 2D occupation grid map: AgRobPP needs a 2D occupation grid map, that is a map composed of grid cells, each cell containing information about free space for robot navigation. In order to obtain an accurate representation on this uneven environment, the map was constructed resorting to the vineyard-segmented 3D PointCloud. The coordinates x_p and y_p of each PointCloud's point are used to construct a 2D occupation grid map. So the grid map cell (x, y) will be:

- occupied, if for every possible altitude of the Point (x_p, y_p) there is at least one point marked as vegetation or wall;
- free, if for every possible altitude of the Point (x_p, y_p) all the points are marked as floor; and
- unknown, if any other situation, for example, if some PointCloud points do not have an associated normal vector.

Smoothing 2D map: The obtained grid map contains unknown cells in well-mapped areas, for example, in the middle of the vineyard's row. When these unknown points are very sparse in free spaces, usually means that they are noise that needs to be cleaned, a smoothing process is applied to

[3]PointCloud Library (PCL) – http://pointclouds.org/.

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

remove them. To remove the noise, each state of the neighbors cells of unknown cells is checked to count the number of free cells or occupied cells, attributing the value of the majority to the unknown cell. If the number of free cells is equal to the number of occupied cells, the unknown cell is marked as an obstacle.

DEM: An important input of this algorithm is the altitude map, which is equivalent to a DEM. The combination of the DEM model and robot's center of mass is needed to find safety trajectories. Although, there are public tools that provide DEMs,[4] they cannot provide with the required resolution for AgRobPP; this resolutions should be at least the same resolution of the occupation grid map. So, a digital elevation map is constructed resorting to the generated PointCloud. Every point in the PointCloud is provided with $(x, y.z)$ coordinates, so, using OpenCV library, a gray-scale image in PNG format is created to store altitude (z) information, considering that image as the altitude map. The minimum value of altitude will be represented with black color and the maximum value with white color, Fig. 15. The z value is normalized according to Eq. (4), where the normalized altitude value is z_{norm}, and z_{min} and z_{max} represent the minimum and maximum registered values of altitude.

$$z_{norm} = \frac{z - z_{min}}{z_{max} - z_{min}} \cdot 255 \tag{4}$$

For interoperability with farm management intelligent systems, PC2GD stores the information in GTIFF format, without being necessary to normalize the altitude value. The GTIFF file is created with Gdallibrary,[5] a format that is also accepted by the presented path planning tool. This way is possible to maintain the compatibility with standard geographic information systems (GIS) and receive an external map from the vineyard's owner and provide a DEM to the farmer as well.

5. Tests and Results

This section presents test and results of AgRobPP with simulated and real data. Section 5.2 presents the operation of AgRobPP with simulated grid and altitude maps. These images were created with GIMP[6] and PC2GD is not used. In Section 5.6, the AgRobPP results are presented with real vineyard data acquired by AgRob V16 and considering PC2GD as described in Section 5.4. Rviz[7] tool was used to visualize the tests, as well to select the initial and target waypoint for the path planning. The robot's dynamics was not simulated; only the generated paths were visualized to validate the proposed concept. The path length is calculated by the sum of distances between every two consecutive waypoints, and the processing time corresponds to the period between the selection of the initial and target waypoints and output path. In these tests, the grid map resolution is 5 cm/cell (square). A smaller resolution would increase the computational complexity. A bigger resolution does not ensure an accurate safety path planning for robot, as it decreases the available space for navigation, which is very limited in vineyards.

5.1. AgRob V16

AgRob V16 is the INESC TEC robotic platform for research and development of robotics technologies for Douro vineyard, Fig. 7. AgRob V16 has four wheels with differential traction configuration. It is equipped with an IMU, Odometry, GNSS, 2D LIDARs and with one camera with special light filters to extract normalized difference vegetation index (NDVI) images.

5.2. Simulated environment

For simulation tests, a simulated steep slope vineyard model is considered,[2] from where an occupation grid map, Fig. 8, and a DEM, Fig. 9, are extracted. In Fig. 8, the black color represents an obstacle, white color represents a free cell and gray color the obstacle expansion. For the altitude map/DEM, depicted in Fig. 9, the black color represents the minimum altitude and the white color the maximum altitude (gradient).

[4]Google Elevation API – https://developers.google.com/maps/documentation/elevation/intro.

[5]Geospatial Data Abstraction Library – http://www.gdal.org/.

[6]GIMP – GNU Image Manipulation Program – https://www.gimp.org.

[7]Rviz – http://wiki.ros.org/rviz.

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

Fig. 7. AgRob V16 robotic platform.

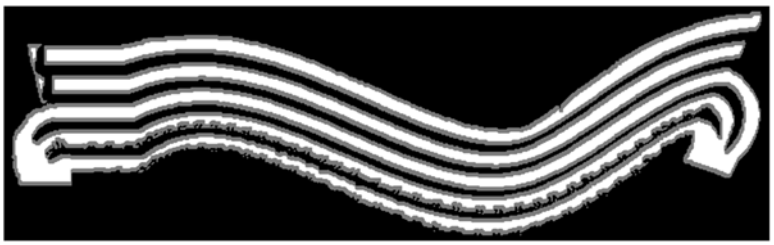

Fig. 8. Occupation grid map of a simulated steep slope vineyard.

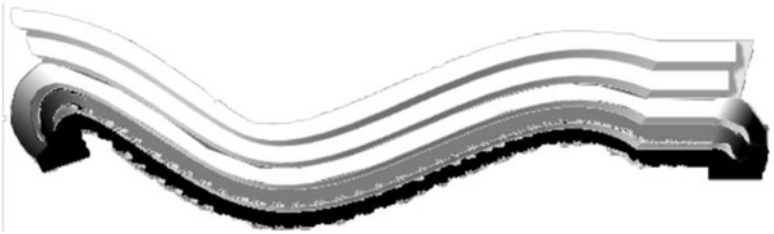

Fig. 9. Simulated altitude map/DEM.

5.3. AgRobPP with simulated maps

AgRobPP extends A*, considering the occupation grid map expanded into 16 layers; this means that each layer will have different information (e.g., more occupied cells due to danger spots for different robot orientations). Figure 10 shows the result for a portion of the simulated vineyard map using the AgRob V16 robot features: dimension 120 × 80 cm, a center of mass C_m (20, 0, 60) cm. The AgRob V16 with an orientation of 180° (yaw) has less free space to navigate safely through the uphill. In this particular scenario the robot's center of mass is not placed into the geometric center of the robot; this will impose different projections of the robot's center of mass in uphills and downhills.

Figure 11 shows the path planning using the extended A* aware of robot's center of mass (AgrobPP) against another A*. Parameters like distance and processing time did not suffer significant changes. The A* without center of mass awareness takes 0.24 s to process and generates a path with 12.10 m length while AgRobPP takes 0.26 s to process and generates a 12.25-m long path. Tables I and II contain specific data about some waypoints of the generated path such as the terrain altitude, robot's orientation in Euler angles (roll, pitch, yaw) and the safety of the path in each waypoint. The

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

Table I. A* detailed results with simulated vineyard without center of mass awareness.

Waypoint number	Altitude (m)	Roll (degrees)	Pitch (degrees)	Yaw (degrees)	Is it a safe zone?
0	12.75	0	0	−90	Yes
1	12.75	0	0	−90	Yes
			...		
11	9.3	−24	13.67	−112.5	Yes
12	7.75	−24.25	13.67	−112.5	Yes
13	6.2	−22.95	15.12	−112.5	Yes
14	4.9	−22.765	12.583	−135	Yes
			...		
23	1.35	−48.95	11.17	157.5	No
			...		
28	0	0	84.01	112.5	No

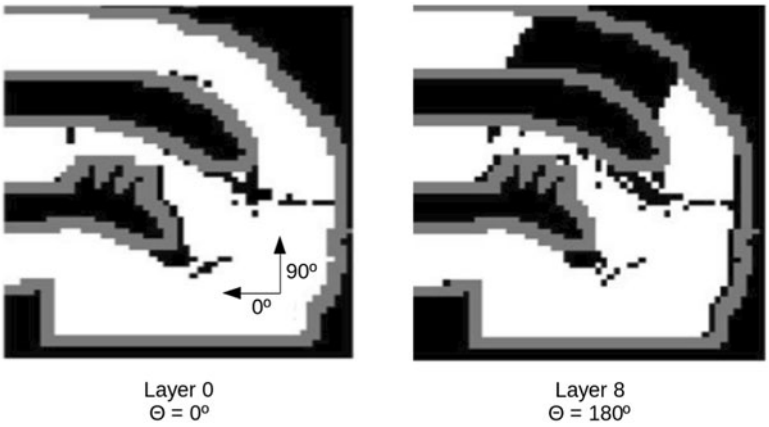

Fig. 10. Occupation grid map after safety verification with center of mass.

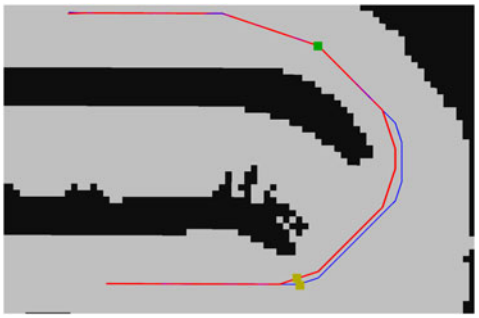

Fig. 11. Simulation tests with A*: red - A* with just orientation; blue – A* aware of center of mass (AgrobPP); green – waypoint number 14; yellow – waypoint number 28.

waypoints in these tables represent the output of AgRobPP, that is, a set of discrete points containing a (x, y, z) position and a (*roll*, *pitch*, *yaw*) orientation. Table I considers a non-safe path obtained by a standard A* algorithm and Table II considers a safer path obtained by AgRobPP (aware of center of mass). From these tables, it is concluded that a standard A* algorithm generates dangerous paths for the robot. From Table I, waypoints 23 and 28 impose a dangerous posture to the robot with high roll/pitch which makes the robot to fall, in Table II closer and safety waypoints for the robot path, with lower values of robot pitch/roll.

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

Table II. A* detailed results with simulated vineyard with center of mass awareness.

Waypoint number	Altitude (m)	Roll (degrees)	Pitch (degrees)	Yaw (degrees)	Is it a safe zone?
0	12.75	0	0	−90	Yes
1	12.75	0	0	−90	Yes
			...		
11	9.3	−24.25	5.28	−112.5	Yes
12	7.75	−24.25	11.11	−112.5	Yes
13	6.2	−22.95	13.67	−112.5	Yes
14	4.9	−22.765	13.67	−112.5	Yes
			...		
23	0.2	−8.33	−2.79	157.5	Yes
			...		
28	0	0	0	135	Yes

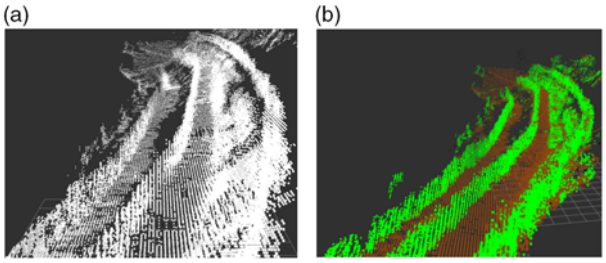

Fig. 12. Vineyard PointClouds. (a) Real vineyard PointCloud; (b) real vineyard-segmented PointCloud; brown – floor; green – vegetation/wall.

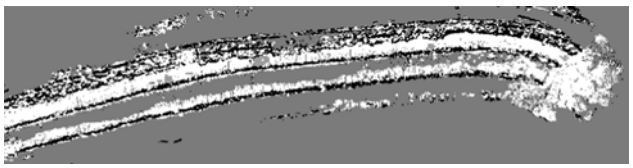

Fig. 13. Raw representation of 2D occupation grid map.

5.4. PointCloud to grid map and DEM

Figure 12(a) shows the available data from the real vineyard, a PointCloud extracted while the robot navigated through just one of the vineyard's rows.

5.4.1. PointCloud segmentation. Once the normal vectors are computed, the *roll* and *pitch* are calculated with the same principle of Eq. (2). Having all the information about the slope of each point, the segmentation will be performed with the attribution of an RGB value to each point. Figure 12(b) shows PointCloud with two class points, the floor points are brown and the vegetation/walls (not floor) points are green.

5.4.2. Generate 2D occupation grid map. Figure 13 shows the generated map, where white color represents a free cell, black an occupied cell and gray a cell with an unknown status of a portion of a real steep slope vineyard, where the robot navigated trough only one of the vineyard's rows. Figure 14 shows the result after the smoothing process.

5.5. Digital elevation model

The Altitude Map/DEM used by the AgropPP, Fig. 15, is a gray-scale image where the minimum registered altitude is represented in black and the maximum altitude is represented in white.

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

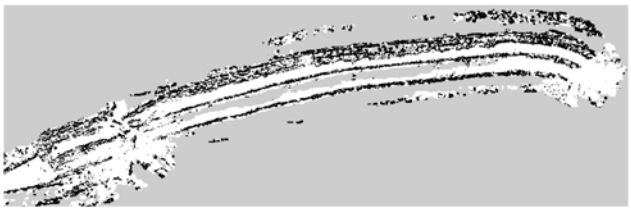

Fig. 14. Smooth representation of 2D occupation grid map.

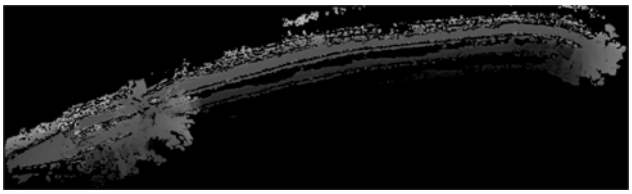

Fig. 15. Altitude map image of real vineyard. Black – minimum altitude; white – maximum altitude.

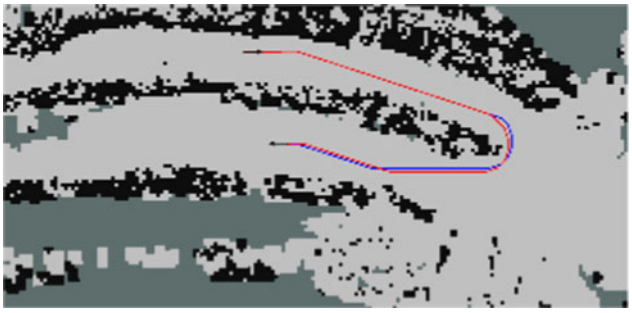

Fig. 16. Real data tests with A*: red – A* with just orientation; blue – A* aware of center of mass (AgrobPP).

5.6. A real data results*

With AgRob V16 robot specification and considering a PointCloud, Fig. 12, obtained with AgRoB V16 on the Quinta do Seixo – Sogrape Vineyard (Pinhão – Portugal), two paths are obtained, Fig. 16. The standard A* path planned (in red) took 0.06 s to process and has 13.6 m of length while AgrobPP path planned(in blue) took 0.06 s and has 16.1 m of length. In Fig. 16 the safer path is represented with blue color and describes a smoother and wider curve performing a safer way of the most challenging task of the navigation in a steep slope vineyard (the transition between vine rows).

Tables III and IV contain specific data about sample waypoints, similar to the tables in Section 5.3. Table III shows proprieties relevant to waypoints for the path not aware of robot's center of mass while Table IV considers a safer path (aware of center of mass – AgRobPP), to prove that standard A* algorithm generates a danger path with high robot pitch/roll while AgRobPP generates paths with lower pitch/roll values.

6. Conclusion

The proposed path planning algorithm (AgRobPP) is aware of the robot's center of mass and ensures a generation of a safe path for any robotic platform, which works in steep slope vineyards.

AgRobPP extends the standard A* algorithm to restrict the maximum robot pitch/roll on the path planned, by considering the characterization of robot's center of mass and terrain slopes (with the DEM terrain). This propriety insures a safe autonomous robot navigation in dangerous and unstructured sloppy terrains. This extension does not increase significantly the AgRobPP computational complexity, observed when compared the processing time of a standard A* with AgRobPP.

Besides PC2GD, from AgRobPP, splits a 3D PointCloud in two more intuitive maps of the vineyard from: a 2D occupation grid map and a high-resolution DEM, compatible with any GIS system. This 2D occupation grid map contains more condensed obstacle information than the traditional 2D-Slam methods, as it is built using a 3D model instead of a single horizontal beam of a LIDAR. The 3D

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

Table III. A* detailed results with real vineyard without center of mass awareness

Waypoint number	Altitude (m)	Roll (degrees)	Pitch (degrees)	Yaw (degrees)	Is it a safe zone?
0	−0.5	0.0	0.0	5	Yes
1	−0.5	0.9	−0.9	0.0	Yes
			...		
5	−0.58	0.9	−0.9	0.0	Yes
6	−0.54	0.9	−0.9	22.5	Yes
			...		
27	−0.7	45	45	90	No
28	−0.8	45	45	112.5	No
			...		
48	−2.8	10.2	−5.6	−180	Yes

Table IV. A* detailed results with real vineyard with center of mass awareness.

Waypoint number	Altitude (m)	Roll (degrees)	Pitch (degrees)	Yaw (degrees)	Is it a safe zone?
0	−0.5	0.0	0.0	5	Yes
1	−0.5	0.9	−0.9	0.0	Yes
			...		
5	−0.58	0.9	−0.9	0.0	Yes
6	−0.54	0.9	−0.9	22.5	Yes
			...		
27	−0.66	0.9	0.9	90	Yes
28	−0.78	0.0	2.3	112.5	Yes
			...		
49	−2.8	10.2	−5.6	−180	Yes

model enables the possibility to build a high-resolution elevation model that is needed by AgRobPP and other components of agricultural ecosystem where the robot is integrated.

As future work, AgrobPP should be optimized in terms of memory usage. Vineyards usually have bigger dimensions than factories/indoor robotic applications. This reflects in bigger occupation grid maps implying an exponential memory usage on the systems. It is necessary to optimize data structure in each cell of the occupation grid map to reduce AgRobPP memory usage. Besides, it will be considered the use of topological maps, which allows to divide a bigger map into several smaller maps. With this feature, AgrobPP will be able to load only a part of the map for the path planning, decreasing the usage of memory and the overall path planning. Other hybrid implementations considering other path planning algorithms (like RRT* or Particle Swarm Optimization) will be considered in the future to coordinate the robot movement with robotic agricultural tasks (e.g., harvesting and pruning). The motion planning and stability analysis based on a dynamic robotic center of mass (i.e., changing the robotic arm position and other implements)[22] will be considered for future work.

Acknowledgments

This work is co-financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation – COMPETE 2020 under the PORTUGAL 2020 Partnership Agreement, and through the Portuguese National Innovation Agency (ANI) as a part of project «ROMOVI: POCI-01-0247-FEDER-017945».

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press

References

1. D. S. O. Correa, D. F. Sciotti, M. G. Prado, D. O. Sales, D. F. Wolf and F. S. Osorio, "Mobile Robots Navigation in Indoor Environments Using Kinect Sensor," *2012 Second Brazilian Conference on Critical Embedded Systems*, Campinas, Brazil (IEEE, 2012) pp. 36–41.
2. F. B. N. dos Santos, H. M. P. Sobreira, D. F. B. Campos, R. M. P. M. dos Santos, A. P. G. M. Moreira and O. M. S. Contente, "Towards a Reliable Monitoring Robot for Mountain Vineyards," *2015 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Vila Real, Portugal (IEEE, 2015) pp. 37–43.
3. S. Rodriguez, X. Tang, J. M. Lien and N. M. Amato (2006, May). "An Obstacle-Based Rapidly-Exploring Random Tree," *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2006*, Orlando, Florida, USA (IEEE) pp. 895–900.
4. M. Pivtoraiko, R. A. Knepper and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *J. Field Rob.* **26**, 308–333 (2009).
5. E. Fernandes, P. Costa, J. Lima and G. Veiga, "Towards an Orientation Enhanced Astar Algorithm for Robotic Navigation," *2015 IEEE International Conference on Industrial Technology (ICIT)*, Seville, Spain (IEEE, 2015) pp. 3320–3325.
6. S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Auton. Rob.* **13**(3), 207–222 (2002).
7. S. Karaman, M. R. Walter, A. Perez, E. Frazzoli and S. Teller, "Anytime Motion Planning Using the RRT," *2011 IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China (IEEE, 2011) pp. 1478–1483.
8. X. Song, X. Fan, Z. Cao and H. Gao, "A TC-RRT-Based Path Planning Algorithm for the Nonholonomic Mobile Robots," *2017 36th Chinese Control Conference (CCC)*, Dalian, China (IEEE, 2017) pp. 6638–6643.
9. F. Auat Cheein, M. Torres Torriti, N. B. Hopfenblatt, Á. J. Prado and D. Calabi, "Agricultural service unit motion planning under harvesting scheduling and terrain constraints," *J. Field Rob.* **34**(8), 1531–1542 (2017).
10. C. Wang, L. Meng, S. She, I. M. Mitchell, T. Li, F. Tung, W. Wan, M. Q.-H. Meng and C. W. de Silva, "Autonomous Mobile Robot Navigation in Uneven and Unstructured Indoor Environments," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada (IEEE, 2017) pp. 109–116.
11. S. Gajjar, J. Bhadani, P. Dutta and N. Rastogi, "Complete Coverage Path Planning Algorithm for Known 2D Environment," *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bengaluru, India (IEEE, 2017) pp. 963–967.
12. J. Valente, D. Sanz, J. Del Cerro, A. Barrientos and M. Á. de Frutos, "Near-optimal coverage trajectories for image mosaicing using a mini quad-rotor over irregular-shaped fields," *Precis. Agric.* **14**(1), 115–132 (2013).
13. J. Jin and L. Tang, "Coverage path planning on three dimensional terrain for arable farming," *J. Field Rob.* **28**(3), 424–440 (2011).
14. T. Goto, T. Kosaka and H. Noborio, "On the Heuristics of A* or A Algorithm in ITS and Robot Path-Planning," *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003,(IROS 2003)*, Las Vegas, USA, vol. 2 (IEEE, 2003) pp. 1159–1166.
15. T. P. Do Nascimento, P. Costa, P. G. Costa, A. P. Moreira and A. G. S. Conceição, "A set of novel modifications to improve algorithms from the A* family applied in mobile robotics," *J. Braz. Comput. Soc.* **19**(2), 167–179 (2013).
16. A. P. Moreira, P. Costa and P. Costa, "Real-Time Path Planning Using a Modified A* Algorithm," **In**: *Proceedings of ROBOTICA 2009-9th Conference on Mobile Robots and Competitions* (2009).
17. L. Santos, F. N. dos Santos, J. Mendes, N. Ferraz, J. Lima, R. Morais and P. Costa, "Path Planning for Automatic Recharging System for Steep-Slope Vineyard Robots," *Iberian Robotics Conference* (Springer, Cham, 2017) (pp. 261–272).
18. R. Schirmer, P. Biber and C. Stachniss, "Efficient Path Planning in Belief Space for Safe Navigation," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2017) pp. 2857–2863.
19. W. Tian, "The Research into Methods of Map Building and Path Planning on Mobile Robots," *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chengdu, China (IEEE, 2017) pp. 1087–1090.
20. T. Stoyanov, M. Magnusson, H. Andreasson and A. J. Lilienthal, "Path Planning in 3D Environments Using the Normal Distributions Transform," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan (IEEE) pp. 3263–3268.
21. J. M. Mendes, F. N. dos Santos, N. A. Ferraz, P. M. do Couto and R. M. dos Santos, "Localization based on natural features detector for steep slope Vineyards," *J. Intell. Rob. Syst.* **93**(3–4), 433–446 (2019).
22. W. Wang, W. Dong, Y. Su, D. Wu and Z. Du, "Development of search and rescue robots for underground coal mine applications," *J. Field Rob.* **31**(3), 386–407 (2014).

https://doi.org/10.1017/S0263574719000961 Published online by Cambridge University Press