ABSTRACT The Agri-Food production requirements need a more efficient and autonomous process, and robotics will play a significant role in this process. Deploying agricultural robots on the farm is still a challenging task. Particularly in slope terrains, where it is crucial to avoid obstacles and dangerous steep slope zones. Path planning solutions may fail under several circumstances, as the appearance of a new obstacle. This work proposes a novel open-source solution called AgRobPP-CA to autonomously perform obstacle avoidance during robot navigation. AgRobPP-CA works in real-time for local obstacle avoidance, allowing small deviations, avoiding unexpected obstacles or dangerous steep slope zones, which could impose a fall of the robot. Our results demonstrated that AgRobPP-CA is capable of avoiding obstacles and high slopes in different vineyard scenarios, with low computation requirements. For example, in the last trial, AgRobPP-CA avoided a steep ramp that could impose a fall to the robot.

INDEX TERMS Agricultural robotics, Bézier curve, collision avoidance, path planning, robotics, steep slope vineyard.

I. INTRODUCTION

The sector of agriculture is crucial for the global economy. With the continuous growth of the world’s population [1], and the decrease in the availability of human resources of agricultural labour [2], the agricultural sector requires a sustainable increase in productivity. The strategic European research agenda for robotics [3] states that robots will have an important role in this task with the automatisation and optimisation of agricultural tasks. A recent literature review under the subject of agricultural robots [4] found that robots have been most explored for harvesting and weeding. The study infers that optimisation and further development is vital. However, the deployment of robots in agriculture is still a challenge, particularly in the context of steep slope vineyards (placed in the Douro Demarcated Region (Portugal), UNESCO Heritage place). Localisation and path planning, which are two crucial tasks to achieve a full autonomous navigation robot, represent several challenges in these scenarios: Global Navigation Satellite Systems (GNSS) gets frequently blocked by the hills providing unstable positioning estimations, and the irregular steep slope terrain presents a challenge for path planning algorithms. To tackle some of these issues, we have proposed VineSlam [5] and Agricultural Robotics Path Planning framework (AgRobPP) [6]. AgRobPP performs path planning operations considering the robot centre of mass and the terrain slopes but requires a complete previous representation of the terrain (map). Such map could be obtained with a Simultaneous Localisation and Mapping (SLAM) process, such as VineSlam, or by resorting to high-resolution aerial images, as proposed in a previous work [7]. VineSlam will be able to provide a full detailed map, but the construction process in extensive agricultural fields would represent a time-consuming or impractical procedure. It becomes easier to extract a map of the entire terrain with satellite images, but the result will never have the same level of detail. Even if a previous high-resolution representation of the terrain were available, unexpected events would be conducive, such as the appearance of new obstacles or changes in the terrain. AgRobPP is able to change the original trajectory with the appearance of new obstacles. Nevertheless, the framework always relies on the previous map information to find an
alternative path. To tackle the problem, this work contribution proposes AgRobPP-CA, an independent collision-avoidance system that relies on real-time local observations (obstacles and elevation) and uses parametric Bézier curves to draw an alternative trajectory. This tool will be used with AgRobPP, and it will proceed to make corrections in the trajectory generated by the path planner whenever necessary. AgRobPP-CA takes into account new obstacles and dangerous inclinations that may not be visible in the map used to generate the initial path. In this paper, Section II presents the related work of collision avoidance strategies used in agricultural contexts. Section III briefly describes the path planning algorithm AgRobPP; and Section IV introduces the collision avoidance tool AgRobPP-CA. Section V reveals the tests and results of AgRobPP-CA under several simulations, and Section VI presents the conclusions of this work.

II. RELATED WORK
The concept of robotic path planning is recurrent in the literature [8]–[10], and consists of finding a collision-free path between an initial and a final point, optimising time, distance, and energy parameters. Path planning may be based on different concepts, such as potential field [11], sampling-based methods [12] and cell decomposition [13]. Independently, path planning algorithms can be divided into two categories: off-line (global) or online (local). Off-line path planners require a previous full environment representation with all the obstacles information, while, in the second category, the map can be constructed/updated during the navigation [8]. A literature review under the topic of robotic path planning in agriculture [14] reveals that most approaches have been resorting to off-line path planning solutions, where it is assumed a full knowledge of the surrounding environment. To make such an assumption may be dangerous for autonomous robot navigation in harsh agricultural terrains, so online methods are preferable.

Zhang et al. [15] proposed an improved Dynamic Window Approach (DWA) to guarantee an optimal collision avoidance system of a global path planning system, transforming an off-line path planning approach into an online method. In the agrarian context, in 2011, Pingzeng et al. [16] presented an obstacle avoidance system based on multi-sensor information, such as ultrasonic, light detection and ranging (LiDAR) and camera. The retrieved data is used to detect the obstacles and their posture. Combined with a fuzzy control algorithm, the agricultural robot navigates automatically in a non-structural environment.

More recently, Juman et al. [17] work approaches the collision avoidance problem with a D* algorithm to navigate through an oil palm plantation autonomously. The trees detection was performed using Viola and Jones framework with a Kinect camera. The planner did not consider all of the robot’s constraints, which led to the incapability of following the generated path in some scenarios.

Fleckenstein et al. [18] approached this topic using a robot with adjustable wheel position. Using a search guidance heuristic, the authors efficiently handled the high number of degrees of freedom in a complex environment with many obstacles.

Ohi et al. [19] resorted to a Dijkstra search algorithm in Voronoi graphs to assure precision pollination with a mobile robot inside a greenhouse. The authors choose a DWA based approach to avoid local obstacles.

A velocity control strategy for collision avoidance was proposed by Xue et al. [20]. The authors performed collision prediction in dynamic environments with an improved obstacle space-time grid map and used a velocity generator for collision avoidance with a cloud model.

An online path planning algorithm was proposed for an automated tractor steering control, following points localised in the ground and using structures provided by the environment for orientation. [21]

Saeed et al. [22] presented a solution for long-term path planning tasks in agriculture. Their study is composed by a method to generate a path to visit all the selected places on a farm, combined with an online path optimisation scheme for re-planning and scheduling the new path at run-time.

In the context of vineyards, an automatic path planning method was studied to perform agricultural tasks in the hilly vineyards of Italy. The maps are generated with Unmanned Aerial Vehicle (UAV) images, and an A* algorithm is used to plan a path between an initial and a goal point. However, the authors did not address the collision-avoidance in the run-time problem [23]. In a similar context, also in the hilly vineyards of Italy, Mammarella et al. [24] describes cooperation between aerial and ground robots, exploiting the DWA based on a receding-horizon scheme to perform local path planning.

Another approach presented a local motion planner in vineyards using an RGB-D camera-based algorithm to generate a proportional control of the robot. A deep learning backup algorithm resilient to illumination variations takes over in case of failure in the RGB-D system. [25]

Ravankar et al. [26] suggest an algorithm that relies on LiDAR data and can avoid dynamic obstacles in a flat vineyard while smoothing the robot’s trajectory.

These presented works in this small literature review, do not address the problem of collision avoidance in steep slope vineyards, considering the robot’s centre of mass, and to the best of our knowledge, such an approach does not exist. Our previous work, AgRobPP [6], is a path planning tool aware of the robot centre of mass based on cell decomposition method with A* search algorithm. It considers a complete knowledge of the environment obstacles and the terrain elevation model, with a slight adaption to change the trajectory with the appearance of new obstacles. AgRobPP may be classified as an online path planning method. Nonetheless, it does not consider changes in the elevation model of the terrain. So, this paper will study a way to perform real-time collision avoidance based on the robot’s sensors local observations, considering the robot’s centre of mass and the real-time perception of the terrain slant.
III. AgRobPP - AGRICULTURAL PATH PLANNING

AgRobPP is an open-source path planning framework built on ROS (Robot Operating System) described in detail in previous works [6], [7], and available at [27]. The framework is composed by three separate tools:

- AgRob Vineyards Detector - a satellite image segmentation tool [7].
- AgRob Topologic - a tool to construct a topological map [7].
- AgRob Path Planning - A path planning approach for uneven terrains [6].

This paper focuses on the AgRob Path Planning tool and intends to improve its capacity of collision avoidance in real-time. AgRob Path Planning tool consists of an A* algorithm for robot navigation in steep slope vineyards characterised by uneven terrain. As Fig. 1 depicts, this algorithm subscribes an Occupancy Grid Map, the Altitude/elevation Map, the robot’s localisation (or starting point) and a goal point. With this data, AgRobPP generates a safe path, considering the robot’s centre of mass and its dimensions, so that the robot does not suffer a large vertical deviation and, consequently, a severe fall. The output provides a safe path described by a set of discrete way-points or by a set of parametric curves. AgRobPP also has a point cloud translator algorithm called PC2GD (Point Cloud to Grid Map and DEM), which extracts the Occupation Grid Map and Digital Elevation Model (DEM) from a point cloud.

IV. AgRobPP-CA - COLLISION AVOIDANCE

The diagram in Fig. 1 mentions a Local Safety Re-Planner used in AgRobPP for local obstacle avoidance. When a new obstacle is detected, it places an order to cancel the current trajectory. Then a goal point is generated to perform a new path planning operation considering the recent obstacle.

This approach has two main drawbacks: changes in the elevation map are not taken into consideration, and real-time performance is not achieved. The A* algorithm search processing time depends upon several factors (i.e. the size of the new obstacle), so it may not be feasible for a real-time collision avoidance task. Nonetheless, the obstacle expansion process (described in [6], [13]) can not be performed instantaneously, as it is executed in the 16 layers of the occupation grid map to consider the different possible headings of the robot.

AgRobPP-CA is an independent framework that uses local observations from the robot’s sensors to detect obstacles and dangerous slopes. This tool is also open-source and is available in [27]. The goal is to perform slight deviations from the original trajectory to reach the destination point in safety. Fig. 2 shows that AgRobPP-CA has three inputs: the LiDAR observations, the parametric curve that robot is following, and the robot’s localisation. The parametric curve may be provided by AgRobPP, but it is not mandatory. Currently, AgRobPP-CA is prepared to work with four wheels differential robots.

AgRobPP-CA is composed by the following steps, which are detailed in the sections below:

- Definition of local safety zone;
- Prior obstacle detection;
- Construction of local elevation map;
- Detection of dangerous slopes;
- Alternative path for collision avoidance.

The results expressed either here, in the step by step demonstration, or in the Section V, were constructed with a Gazebo simulation. The robot has a four wheel

1http://gazebosim.org/
differential configuration, and represents a simulated version of AgRobv16 [28] shown in Fig. 3, equipped with a VLP-16 Puck LiDAR. The robot navigates autonomously using a trajectory controller software based on parametric curves, which was previously developed by Pinto et al. [29]. In this section, the environment is composed by a simulation of a steep slope vineyard created with a modelling software, Fig. 4 [30].

The framework contains different coordinate frames, namely the Map frame $M$, the Sensor frame $S$, the Base frame $B$, and the Footprint frame $F$. The frames location is illustrated in Fig. 5.

The map frame $M$ is fixed in the environment, assuming that the real terrain is stationary to this frame. The base sensor $B$ is attached to the centre of the robot’s base, and is related to the map frame $M$ through a translation $\tau_{mb}$ and a rotation $\omega_{mb}$. This transformation is provided by the localisation system, which, for this particular scenario is given by Gazebo’s ground truth. The remaining relations between frames are static and expressed in equation (1). The frames $F$ and $S$, are related to the frame $B$ without any rotation and the respective static translations: $\tau_{bs}$, $\tau_{bf}$.

$$\begin{align*}
\tau_{r_1, r_2} &= \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ meters} \\
\tau_{b,s} &= \begin{bmatrix} 0.3 \\ 0.0 \\ 0.92 \end{bmatrix} \ \\
\tau_{b,f} &= \begin{bmatrix} 0.0 \\ 0.0 \\ -0.132 \end{bmatrix} 
\end{align*}$$

A. SAFETY REGION DEFINITION

The initial step is to define the region around the robot that will be considered for collision avoidance purposes. At this stage, characteristics of the robot must be provided as well, such as the robot’s whole dimension, footprint and centre of mass. The region around the robot is rectangular, and the measurements are provided in meters. Given this data, AgRobPP-CA defines a local grid with a previously provided resolution. The example presented in Fig. 6 shows the LiDAR observations in a square safety region with 100 $m^2$ around the robot frame $B$, represented in a grid with 5cm/pixel resolution.

B. PRIOR OBSTACLE DETECTION

The initial obstacle detection is simply based on the local observation of the LiDAR. So, the LiDAR measurements close to the robot conducive to collision are marked as obstacles. Considering the referential cantered in the robot’s base, frame $B$, every $z$-axis measurement between zero and the robot’s height is viewed as an obstacle, as long as the planar euclidean distance to the robot is less than 3 meters. Fig. 7 shows the simulation of the robot with a vineyard and an obstacle and its representation in the safety region. The LiDAR observations marked as potential barriers are identified with the yellow colour. In this figure, the safety map classifies the ramp access to the vineyard as a potentially dangerous zone. With the construction of the local elevation, in the posterior stage, all obstacles are analysed to verify if they actually represent a collision or fall danger.

---

2 https://velodynelidar.com/products/puck/
C. LOCAL ELEVATION MAP CONSTRUCTION

The elevation map is constructed with continuous observations of the LiDAR. This part requires a localisation system, which is often a requirement for any global path planning system. The elevation map is saved into a grid centred in the robot frame with the z-axis information in the map frame \( M \). The pseudo-code of this process is presented in Algorithm 1. For this pseudo-code, \( L \) represents a data structure in a list format, and \( G \) is the matrix where the elevation map is saved.

Algorithm 1 Local Elevation Map Process

1: Read all points captured by the LiDAR in the frame \( S \)
2: Transform points from frame \( S \) to \( M \): \( \omega^{-1}_b s, \omega^{-1}_m \tau_{mb} \)
3: For all points \( \in \) frame \( M \):
   - if point \( \in \) L then
     discard point
   - else add point to \( L \)
4: For all points \( \in \) L:
   - Transform point from frame \( M \) to \( B \): \( \omega_{mb} \tau_{mb} \)
   - if (point \( \in \) frame \( B \)) is outside limits of \( G \) then
     remove point from \( L \) and from \( G \)
   - else add point to \( G \) with z-axis \( \in \) frame \( M \)

The algorithm 1 is activated when the robot moves, running in a loop at a pre-defined rate in ROS. We must ensure that the rate of such a loop is sufficient to run the entire process at a maximum robot speed with a specific grid resolution. Considering the resolution of the grid elevation map \( \text{map}_{\text{res}} = 0.05 \) m, we should assure not only that the code is executed in a time that allows capturing information every 0.05 m, but also that the code is not executed so fast that the information will be repeated in several iterations.

Considering a speed \( v_{\text{robot}} = 0.5 \) m/s, the ROS loop must have a duration of at least \( \frac{\text{map}_{\text{res}}}{v_{\text{robot}}} \) seconds, that is, a rate of 10 Hz. Considering this rate of 10 Hz, the execution of the loop needs to occur in every iteration. At half of the speed, \( v_{\text{robot}} = 0.25 \) m/s, the algorithm only has to be executed each \( \frac{\text{map}_{\text{res}}}{v_{\text{robot}}} \times \text{Rate iterations} \), that is, every 2 iterations.

Fig. 8 shows the result of a local elevation map constructed while the robot travelled through the trajectory marked with the orange line, at a speed of 0.25 m/s with a 0.05 m resolution. In this visualisation, the white colour represents the maximum altitude, while the black represents zero elevation.

D. DANGEROUS SLOPE DETECTION

During the autonomous robot navigation process, the remaining trajectory and its surroundings will constantly be verified to detect either dangerous slopes unknown by the robot and to detect false obstacles previously marked in the initial stage of the collision avoidance process.

The process of detecting a dangerous slope uses the information from the local elevation map to estimate the terrain inclination. It verifies the robot centre of mass projection to avoid a large vertical deviation. The method is detailed in a previous work [6] applied to a global map.

Algorithm 2 presents a brief description of the procedure. Here \( C \) represents the input parametric curve that the robot is following. \( dC \) is a set of points which contains the discrete representation of \( C \), in the base frame \( B \). The input parametric curve follows the format expressed in equation (2), where \( n \) represents the degree of the curve. We are considering a maximum of 4th degree curves.

\[
P(t) = \sum_{i=0}^{n} c_{n-i} t^{n-i}, \quad t \in [0, 1]
\]

\[
P(t) = c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0, \quad t \in [0, 1] \quad n = 4
\]

E. COLLISION AVOIDANCE METHOD

The problem is structured into a state machine, Fig. 9, with four states: \text{WAITING\_FOR\_PATH}, \text{FOLLOW\_ORIGINAL\_PATH},...
**RECALCULATE_PATH** and **FOLLOW_ALTERNATIVE_PATH**. In the initial state, **WAITING_FOR_PATH**, the robot waits for the reception of a parametric curve and is stopped. Once the input is received, the robot starts following it, changing the state to **FOLLOW_ORIGINAL_PATH**. Now, either the robot completes the trajectory and returns to the initial state or finds an obstacle ahead and goes to the **RECALCULATE_PATH** state. Here, the framework tries to design an alternative safe path to contour the obstacle. If there is no success, the robot stops and returns to the initial state. When an alternative is found, the machine follows the **FOLLOW_ALTERNATIVE_PATH** state. Once the new path gets travelled, the tool goes to **FOLLOW_ORIGINAL_PATH**, unless a different obstacle is detected. In this case, the **RECALCULATE_PATH** is called again. At any moment, when the user or the robot’s controller software send a stop signal, the state machine is rebooted into its initial state.

The collision avoidance procedure that occurs in the state **RECALCULATE_PATH** is performed with a heuristic approach using parametric Bézier curves (Equation (3)). The Bézier representation is helpful to define a curve based on several control points. Nevertheless, the robot’s software controller reads the simpler parametric form, so the same curve is later converted to the format of Equation (2). Every alternative path is composed of a 4th degree Bézier curve (3).

\[
B(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i}t^i P_i, \quad t \in [0, 1]
\]

\[
B(t) = (1-t)^4P_0 + 4t(1-t)^3P_1 + 6t^2(1-t)^2P_2 + 4t^3(1-t)P_3 + t^4P_4, \quad t \in [0, 1] \quad n = 4
\]  

(3)

Fig. 10 depicts the generation of an alternative path using a Bézier curve. The original trajectory is designed with a red colour. The control points \( P_0 \) to \( P_4 \), from Equation (3), are represented in the picture with green circles. The yellow circle denotes an obstacle, and the red point \( P_e \) stands for the collision point. \( P_m \) represents the middle point of the new path, and its location is crucial to avoid a possible collision.

The resulting alternative trajectory is illustrated with green colour. The definition of the controls points of the new Bézier curve obeys the following procedure:

- \( P_0 \) is placed in the current robot position, that is, in the origin of base frame \( B \).
- \( P_e \) is the trajectory point intersected by the obstacle.
- \( \text{Length 1} \) is the linear distance from \( P_0 \) to \( P_e \) (over the trajectory).
- \( \text{Length 2} = \text{Length 1} \) and is the linear distance from \( P_e \) to \( P_4 \) (over the trajectory).
- \( P_4 \) is placed on the original trajectory at \( P_e + \text{Length 2} \).
- \( P_1 \) is placed in a fixed distance \( \text{dist}(P_0, P_4)/4 \) after \( P_0 \) following the direction of the curve (6).
- \( P_3 \) is placed with the same fixed distance \( \text{dist}(P_0, P_4)/4 \) before \( P_4 \) following the direction of the curve (6).

The control point \( P_m \) is calculated with the point \( P_m \), which is iterative. Given the Equation (4) to define the alternative parametric Bézier curve, we consider the point \( P_m \) will be in the middle of the trajectory, as \( \text{Length 1} = \text{Length 2} \).

\[
B_x(t) = (1-t)^4P_{0x} + 4t(1-t)^3P_{1x} + 6t^2(1-t)^2P_{2x} + 4t^3(1-t)P_{3x} + t^4P_{4x}, \quad t \in [0, 1]
\]

\[
B_y(t) = (1-t)^4P_{0y} + 4t(1-t)^3P_{1y} + 6t^2(1-t)^2P_{2y} + 4t^3(1-t)P_{3y} + t^4P_{4y}, \quad t \in [0, 1]
\]  

(4)

With such assumption, for the point \( P_m \), the variable \( t \) of equation (4) is 0.5. Then, the control point \( P_m \) will be extracted from equation (5). \( P_0 \) and \( P_4 \) are already defined. \( P_1 \) and \( P_3 \) are defined in equation (6), where \( A_0 \) and \( A_4 \) are the angles of the trajectory heading in the respective points \( P_0 \) and \( P_4 \). \( D_1 \) and \( D_3 \) represent the respective distances: \( P_0, P_1 \) and \( P_3, P_4 \).

\[
\begin{align*}
P_m &= 0.0625 \cdot P_0 + 0.25 \cdot P_1 + 0.375 \cdot P_2 + 0.0625 \cdot P_3 \\
& \quad + 0.25 \cdot P_3 + 0.0625 \cdot P_4 \\
P_m &= 0.0625 \cdot P_0 + 0.25 \cdot P_1 + 0.375 \cdot P_2 + 0.0625 \cdot P_3 \\
& \quad + 0.25 \cdot P_3 + 0.0625 \cdot P_4
\end{align*}
\]  

(5)
\[
\begin{align*}
P_{1x} &= P_0x + \cos(A_{P_0})D_1, \\
P_{1y} &= P_0y + \sin(A_{P_0})D_1,
\end{align*}
\]
\[
\begin{align*}
P_{3x} &= P_{4x} - \cos(A_{P_4})D_3, \\
P_{3y} &= P_{4y} - \sin(A_{P_4})D_3.
\end{align*}
\]

(6)

To determine all the necessary points, a value will be attributed to the point \( P_m \), following the below iterative method.

1) Create a perpendicular vector to the original trajectory in the point \( P_c \) with length \( P_L \) (initially \( P_L = \) robot’s width).
2) Place \( P_m \) in the extremity of the vector, and use the value to extract \( P_2 \) from equation (5).
3) Generate a new Bézier curve, and perform a safety verification in the previously created local grid safety region.
4) If the first curve is not valid rotate the vector (180°) and repeat the process.
5) If both options are not valid, increment \( P_L \) and repeat the whole process until one of the curves is valid, or the euclidean distance between \( P_m \) and \( P_c \) becomes greater than 2 meters. We consider this situation to be the maximum deviation allowed.

If a valid trajectory is achieved, the Bézier curve is converted to the parametric format of equation (2), according to the formulation in (7).

\[
P(t) = t^4(P_0 - 4P_1 + 6P_2 - 4P_3) \\
+ t^3(-4P_0 + 12P_1 - 12P_2 + 4P_3) \\
+ t^2(6P_0 - 12P_1 + 6P_2) \\
+ t(4P_1 - 4P_0) + P_0, \quad t \in [0, 1]
\]

(7)

Finally, the curve is ready to be sent to the robot. However, we need to recover the remaining of the original path. With the initial trajectory defined in the format of equation (2), it is known that the transition from the obstacle avoidance path to the original one is performed in the point \( P_4 \), which corresponds to \( t = t_{p_4} \), a known value. Then, the equation is transformed so that \( t \in [t_{p_4}, 1] \), which is achieved with the following change: \( t = t_{p_4} + (1 - t_{p_4})t \). By applying this variable shift, the resulting equation (8) expresses the remaining of the initial trajectory from the point \( P_4 \) to the final point. To simplify the visualisation of equation (8), \( k = (1 - t_{p_4}) \).

\[
P_r(t) = t^4c_4k^4 + t^3(4c_4t_{p_4}k^3 + c_3k^3) \\
+ t^2(6c_4t_{p_4}^2k^2 + 3c_3t_{p_4}k + c_2k^2) \\
+ t(4c_4t_{p_4}^3k + 3c_3t_{p_4}^2k + 2c_2t_{p_4}k + c_1k) \\
+ c_4t_{p_4}^4 + c_3t_{p_4}^3 + c_2t_{p_4}^2 + c_1t_{p_4} + c_0, \quad t \in [0, 1]
\]

(8)

Once the curves (7) and (8) are sent to the trajectory controller, the robot is able to contour the obstacle and reach its destiny.

V. TESTS AND RESULTS

The tests of AgRobPP-CA were performed with the simulated version of the robot AgRobv16 (Fig. 3) in different simulated environments, such as empty space, straightforward flat vineyard, curved vineyard and simulation of an actual steep slope vineyard (Quinta do Seixo). In all trials, the trajectory is manually designed. For all the pictures portraying AgRobPP-CA results, the robot AgRobv16 is following either an orange or a red line. The orange line corresponds to the original trajectory, while the red line portrays the alternative path found by AgRobPP-CA. The pictures also illustrate blue circles which represent the beginning and end of the trajectory, and the LiDAR observations with several colour points.
The basic scenario expressed in Fig. 11 consists of open space with one and two simple obstacles. The application of AgRobPP-CA is shown in Fig. 12, where the LiDAR observation of the obstacles is visible with the green points.

The second test scenario (Fig. 13) consists of a simulation of a simple vineyard with a human being in the middle of the row as an obstacle. The local observations of AgRobPP-CA are shown in Fig. 14 with the visualisation of the safety region points and the representation of the local elevation map. The resulting obstacle avoidance appears in Fig. 15. The larger yellow circle represents the estimated collision point. The third scenario is similar, with a curved vineyard depicted in Fig. 16. The result is expressed in Fig. 17.
The last test scene (Fig. 18) represents a simulation of “Quinta do Seixo”, a real steep slope vineyard located in the Douro region of Portugal (41.166775, −7.555080). The simulated scenario is depicted in Fig. 19.

The first test demonstrates a path placed on an uphill trail blocked by an obstacle. The Gazebo simulation is visible in Fig. 20. The visualisation of the trajectory and the obstacle avoidance from AgRobPP-CA is presented in Fig. 21. As the larger yellow circle represents the expected collision point, the figures indicate that AgRobPP-CA performed a small deviation in an initial phase. However, with the continuous information acquisition from the sensors, a new collision point is estimated, forcing AgRobPP-CA to perform a bigger deviation.

In the second trial, the path was purposely designed on top of a dangerous slope without any artificial obstacles. The robot AgRobv16 can not perform this trajectory safely, as Fig. 22 demonstrates.

With AgRobPP-CA activated, a new and safe path is generated. The process is similar to Fig. 21. Starting with a small contour of the dangerous zone, and increasing the deviation with the acquisition of new information, Fig. 23.

The corresponding Gazebo simulation is visible in Fig. 24, with an available video demonstration of these two situations in Zenodo.

https://doi.org/10.5281/zenodo.6169082
FIGURE 26. Velocity plot of AgRobv16 performing a dangerous trajectory along time with AgRobPP-CA. Blue - Linear Velocity (m/s); Red - Angular Velocity (s⁻¹).

FIGURE 27. Z-axis pose (m) plot of AgRobv16 performing a dangerous trajectory along time without AgRobPP-CA.

FIGURE 28. Z-axis pose (m) plot of AgRobv16 performing a dangerous trajectory along time with AgRobPP-CA.

FIGURE 29. Euler angles plot of AgRobv16 performing a dangerous trajectory along time without AgRobPP-CA. Red - Yaw; Blue - Pitch; Grey - Roll.

FIGURE 30. Euler angles plot of AgRobv16 performing a dangerous trajectory along time with AgRobPP-CA. Red - Yaw; Blue - Pitch; Grey - Roll.

The plots below show the robot’s velocities and poses along time while performing the dangerous trajectory from Fig. 22. Fig. 25 and 26 present the linear and angular velocity of the robot without and with AgRobPP-CA. The results show greater stability when AgRobPP-CA is activated (Fig. 26).

The robot’s pose is plotted just in the Z-axis for better resolution, as this is the most relevant value to observe when the robot is trying to approach a ramp. These plots are visible in Fig. 27 and 28. It is visible that without AgRobPP-CA the robot can not reach the maximum altitude as it does not overcome the steep ramp.

The robot’s orientation is represented by the respective Euler angles (roll, pitch and yaw) in radians. Fig. 29 and 30 plot these values indicating that the robot is safer with AgRobPP-CA. Fig. 29 shows that the robot’s pitch approximates to 1 rad (≈ 57 degrees), which is dangerous and may impose a fall of the platform. With AgRobPP-CA, Fig. 30, the worst registered case is a peak at approximately 0.5 rad (≈ 28 degrees).

VI. CONCLUSION

The current paper presents an approach for real-time collision avoidance in steep slope agricultural fields based on local observations. For such purpose, we have proposed AgRobPP-CA. This tool constantly verifies the current robot’s trajectory looking for predictable collisions or dangerous inclined zones as AgRobPP-CA considers the robot’s centre of mass. The local observations consist of the LiDAR measurements for obstacle and harmful slope detection. Once a collision is detected, AgRobPP-CA uses an iterative based approach to generate an alternative path with parametric Bézier curves. The method was tested in different simulated scenarios, including a straight vineyard, a curved vineyard, and an actual steep slope vineyard (“Quinta do Seixo”). The results demonstrated that the tool is capable of avoiding obstacles in all scenarios. In the tests performed with the steep slope vineyard, AgRobPP-CA generated an alternative path to avoid a dangerous slope. As future work, we will carry on the tests to an actual vineyard. Other sensors, such as OAK—D camera⁴ with a deep learning model to detect people and animals, will be considered. A new LiDAR (Livox MID-70 ⁵) will be pointed to the floor to improve the perception of the elevation map.

⁴https://store.opencv.ai/products/oak-d
⁵https://www.livoxtech.com/mid-70
REFERENCES

[1] M. Perry, “Science and innovation strategic policy plans for the 2020s (EU, AU, UK): Will they prepare us for the world in 2050?”, Appl. Econ. Finan., vol. 2, no. 3, pp. 76–84, 2015.

[2] M. Leschcheva, A. Ivolga, J. Sabić, B. Kuzman, and A. J. Vasile, “Human resources for agricultural organizations of agro-industrial region, areas for improvement,” in Proc. Sustain. Agricult. Rural Develop. Terms Repub. Serbia Strategic Goals Realization within Danube Region, Support Programs Improvement Agricult. Rural Develop., Belgrade, Serbia, 2017, pp. 14–15.

[3] E. Robotics. (2014). Strategic Research Agenda for Robotics in Europe 2014–2020. Accessed: Apr. 21, 2018. [Online]. Available: https://www.eu-robotics.net/cms/upload/topic_groups/SRA2020_SPARC.pdf

[4] S. Fountas, N. Mylonas, I. Malounas, E. Rodias, C. Hellmann Santos, and E. Pekkeriet, “Agricultural robotics for field operations,” Sensors, vol. 20, no. 9, p. 2672, May 2020.

[5] A. S. Aguiar, F. N. dos Santos, H. Sobreira, J. B. Cunha, and A. J. Sousa, “Particle filter refinement based on clustering procedures for high-dimensional localization and mapping systems,” Robot. Auto. Syst., vol. 137, Mar. 2021, Art. no. 103725.

[6] L. Santos, F. Santos, J. Mendes, P. Costa, J. Lima, R. Reis, and P. Shinde, “Path planning aware of robot’s center of mass for steep slope vineyards,” Robotica, vol. 38, no. 4, pp. 684–698, 2020.

[7] L. C. Santos, A. S. Aguiar, F. N. Santos, A. Valente, and M. Petry, “Occupancy grid and topological maps extraction from satellite images for path planning in agricultural robots,” Robotics, vol. 9, no. 4, p. 77, Sep. 2020.

[8] P. Raja and S. Pugazhenithi, “Optimal path planning of mobile robots: A review,” Int. J. Phys. Sci., vol. 7, no. 9, pp. 1314–1320, Feb. 2012.

[9] E. Galceran and M. Carreras, “A survey on coverage path planning for robotics,” Robot. Auton. Syst., vol. 61, no. 12, pp. 1258–1276, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092188901301677X

[10] T. T. Mac, C. Cotop, D. T. Tran, and R. De Keyser, “Heuristic approaches in robot path planning: A survey,” Robot. Auto. Syst., vol. 86, pp. 13–28, Dec. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889015300671

[11] M. Pivtoraiko, R. A. Knepper, and A. Kelly, “Differentially constrained mobile robot motion planning in state lattices,” J. Field Robot., vol. 26, no. 3, pp. 308–333, Mar. 2009.

[12] T. T. Mac, C. Cotop, D. T. Tran, and R. De Keyser, “Heuristic approaches in robot path planning: A survey,” Robot. Auto. Syst., vol. 86, pp. 13–28, Dec. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889015300671

[13] M. Pivtoraiko, R. A. Knepper, and A. Kelly, “Differentially constrained mobile robot motion planning in state lattices,” J. Field Robot., vol. 26, no. 3, pp. 308–333, Mar. 2009.

[14] F. O. Coelho, J. P. Carvalho, M. F. Pinto, and A. L. Marcato, “Direct-DRRT*: A RRT improvement proposal,” in Proc. 13th APCA Int. Conf. Control Soft Comput. (CONTROLO), Jun. 2018, pp. 154–158.

[15] E. Fernandes, P. Costa, J. Lima, and G. Veiga, “Towards an orientation enhanced Astar algorithm for robotic navigation,” in Proc. IEEE Int. Conf. Ind. Technol. (ICIT), Mar. 2015, pp. 3320–3325.

[16] L. C. Santos, F. N. Santos, E. J. Solteiro Pires, A. Valente, P. Costa, and S. Magalhaes, “Path planning for ground robots in agriculture: A short review,” in Proc. IEEE Int. Conf. Auto. Robot. System Competitions (ICARSC), Apr. 2020, pp. 61–66.

[17] F. Zhang, N. Li, T. Xue, Y. Zhu, R. Yuan, and Y. Fu, “An improved dynamic window approach integrated global path planning,” in Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO), Dec. 2019, pp. 2873–2878.

[18] L. Pingzeng, B. Shusheng, Z. Guansheng, W. Wenshan, G. Yushu, and D. Zhenmin, “Obstacle avoidance system for agricultural robots based on multi-sensor information fusion,” in Proc. Int. Conf. Comput. Sci. Netw. Technol., Dec. 2011, pp. 1181–1185.

[19] M. A. Juman, Y. W. Wong, R. K. Rajkumar, and C. Y. H’ng, “An integrated path planning system for a robot designed for oil palm plantations,” in Proc. IEEE Region 10 Conf. (TENCON), Nov. 2017, pp. 1048–1053.

[20] F. Fleckenstein, C. Dornhege, and W. Burgard, “Efficient path planning for mobile robots with adjustable wheel positions,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2017, pp. 2454–2460.

[21] N. Oh, K. Lassak, R. Watson, J. Strader, Y. Du, C. Yang, G. Hedrick, J. Nguyen, S. Harper, D. Reynolds, C. Kilic, J. Hikes, S. Mills, C. Castle, B. Buzzo, N. Waterland, J. Gross, Y.-L. Park, X. Li, and Y. Gu, “Design of an autonomous precision pollination robot,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Oct. 2018, pp. 7711–7718.

[22] J. Xue, C. Xia, and J. Zou, “A velocity control strategy for collision avoidance of autonomous agricultural vehicles,” Auto. Robots, vol. 44, no. 6, pp. 1047–1063, Jul. 2020.

[23] M. Pichler-Schedler, R. Ritter, C. Lindinger, R. Amsterstorfer, and R. Edelbauer, “Path planning for semi-autonomous agricultural vehicles,” in Reinventing Mechatronics: Developing Future Directions for Mechatronics, X.-T. Yan, D. Bradley, D. Russell, and P. Moore, Eds. Cham, Switzerland: Springer, 2020, pp. 35–53, doi: 10.1007/978-3-030-29134-0_4.

[24] R. A. Saeed, G. Tomasi, G. Govindarajan, R. Vidoni, and K. D. Von Ellenrieder, “Metrology-aware path planning for agricultural mobile robots in dynamic environments,” in Proc. IEEE Int. Workshop Metrol. Agricult. Forestry (MetroAgriFor), Nov. 2021, pp. 448–453.

[25] J. Zoto, M. A. Musci, A. Khaliq, M. Chiaberge, and I. Aicardi, “Automatic path planning for unmanned ground vehicle using UAV imagery,” in Advances in Service and Industrial Robotics, K. Berns and D. Görge, Eds. Cham, Switzerland: Springer, 2020, pp. 223–230.

[26] M. Mammarella, L. Comba, A. Biglia, F. Dabbene, and P. Gay, “Cooperative agricultural operations of aerial and ground unmanned vehicles,” in Proc. IEEE Int. Workshop Metrol. Agricult. Forestry (MetroAgriFor), Nov. 2020, pp. 224–229.

[27] D. Aghi, V. Mazzia, and M. Chiaberge, “Local motion planner for autonomous navigation in vineyards with a RGB-D camera-based algorithm and deep learning synergy,” in Autonomous and safe navigation of mobile robots in vineyard with smooth collision avoidance,” Agriculture, vol. 11, no. 10, p. 954, Sep. 2021.

[28] (2020). Agrob_pp–Agricultural Robot Path Planning. Accessed: Feb. 12, 2021. [Online]. Available: https://gitlab.inesctec.pt/agrob/agrob_pp

[29] A. S. Pinto de Aguiar, M. A. Riem de Oliveira, E. F. Pedrosa, and F. B. Neves dos Santos, “A camera to LiDAR calibration approach through a camera to LiDAR calibration approach,” in Proc. IEEE Int. Workshop Mechatrol. Agricult. Forestry (MetroAgriFor), Nov. 2021, pp. 444–450.

[30] A. J. Sousa, “Particle filter refinement based on clustering procedures for path planning,” in Proc. 13th APCA Int. Conf. Control Soft Comput. (CONTROLO), Jun. 2018, pp. 154–158.

[31] E. Fernandes, P. Costa, J. Lima, and G. Veiga, “Towards an orientation enhanced Astar algorithm for robotic navigation,” in Proc. IEEE Int. Conf. Ind. Technol. (ICIT), Mar. 2015, pp. 3320–3325.

[32] L. C. Santos, F. N. Santos, E. J. Solteiro Pires, A. Valente, P. Costa, and S. Magalhaes, “Path planning for ground robots in agriculture: A short review,” in Proc. IEEE Int. Conf. Auto. Robot. System Competitions (ICARSC), Apr. 2020, pp. 61–66.

[33] F. Zhang, N. Li, T. Xue, Y. Zhu, R. Yuan, and Y. Fu, “An improved dynamic window approach integrated global path planning,” in Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO), Dec. 2019, pp. 2873–2878.