Path Planning for Automatic Recharging System for Steep-Slope Vineyard Robots

Luís Santos[1], Filipe Neves dos Santos[2](✉), Jorge Mendes[2], Nuno Ferraz[2], José Lima[2,3], Raul Morais[2,4], and Pedro Costa[1,3]

[1] Faculty of Engineering, University of Porto, Porto, Portugal
{ee12155,pedrogc}@fe.up.pt
[2] INESC TEC - INESC Technology and Science, Porto, Portugal
{fbsantos,jorge.m.mendes,nuno.a.ferraz}@inesctec.pt, jllima@ipb.pt, rmorais@utad.pt
[3] Polytechnic Institute of Bragança, Bragança, Portugal
[4] Universidade de Trás-os-Montes e Alto Douro, Vila Real, Portugal

Abstract. Develop cost-effective ground robots for crop monitoring in steep slope vineyards is a complex challenge. The terrain presents harsh conditions for mobile robots and most of the time there is no one available to give support to the robots. So, a fully autonomous steep-slope robot requires a robust automatic recharging system. This work proposes a multilevel system that monitors a vineyard robot autonomy, to plan offline the trajectory to the nearest recharging point and dock the robot on that recharging point considering visual tags. The proposed system called VineRecharge was developed to be deployed into a cost-effective robot with low computational power. Besides, this paper benchmarks several visual tags and detectors and integrates the best one into the VineRecharge system.

1 Introduction

The strategic European research agenda for robotics [1] states that robots can improve agriculture efficiency and competitiveness. However, few commercial robots for agricultural applications are available [2]. In Europe space few European funded projects are developing monitoring robots for flat vineyards: the VineRobot [3] and Vinbot [4]. However, vineyards built on steep slope hills presents an higher complex environment for the machinery and automation development. These called steep slope vineyards exist in Portugal in the Douro region - an UNESCO heritage place - Fig. 1, and in other regions of five European countries. The context of a vineyard built in a steep hill presents several robotics challenges for the localization, path-planning, safety, perception systems, and automatic energy management [5].

This paper benchmarks different visual tags and detectors, in terms of accuracy and performance, and presents a system called VineRecharge to be deployed into cost-effective robots with low computational power for automatic energy management. In this paper, Sect. 2 presents the related work to

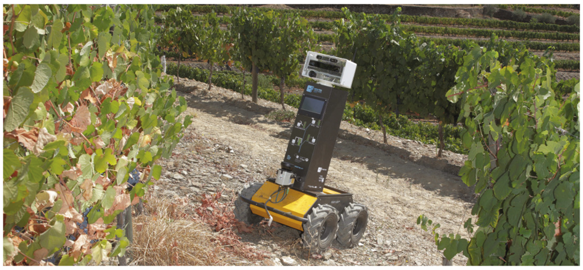


Fig. 1. AgrobV16 working in steep slope terraced vineyard in the douro region of Portugal.

the automatic recharging systems problem and relative localization. Section 3 presents the VineRecharge architecture and its main components. Section 4 presents the benchmarking of six visual tags detector and results obtained by the VineRecharge system. Section 5 presents the paper conclusions.

2 Related Work

A fully autonomous robot requires a docking capability for self energy charging. However, for the docking capability is required to develop navigation algorithms to drive the robot towards the docking station, precise relative positioning systems for system coupling and mechanical interface for energy transference and data communication. Several works related to the robotic docking systems can be found. [6] presents an autonomous docking system with integrated algorithms for mobile self-reconfigurable robots equipped with inexpensive sensors, such as infrared emitter-receiver pairs (IR) and encoders.

For a successful docking, GPS, Vision, and RFID based techniques can be used as well to guide the robot to a near docking station, but at the moment there has not been great success using this kind of technology because the rate error and is susceptibility to interferences. But [7] proposes a system based of RFID, using a bi-directional antenna that will communicate with and RF transponder, enabling the robot to identify the transponder and decide which is the best path to follow by comparing the signal strength. Although this approach has been successful in indoor environments without major obstacles and proved that the system can find the target without real time scanning, in real environment their effectiveness may be affected, because the existence of interferences can corrupt the signal used to guide the robot. In contrast, to the previous works in [8] is proposed a magnetic alignment-based approach to solve underwater docking system. Considering magnetic alignment-based, [9] proposes a multisensory docking strategy, which includes visual-sensor guided rough positioning, Hall-sensor-guided fine positioning, and the locking between moving and target modules, guarantees robust docking without sacrificing reconfigurability. [10] presents a model that exceeds the systems adaptability of the problem and the low error tolerance, the model incorporates magnets in both the docking station and the robot module and by using the magnetic force generated between them ensures a robust docking without any error.

Visual landmarks are being widely used to increase the localization accuracy and has high potential for docking systems [11–16]. For example QR-based landmarks have been used extensively because of their capability to store large amounts of information and their resistance against distortion and damage [13]. Many visual landmark models are designed in consideration of detection mechanism. Some examples of these landmarks are: ARTags, AprilTags, QRCodes, Barcodes, ArUco markers, and ARToolKit. These landmarks are benchmarked in Sect. 4.1. [17] describes a new localization method for an indoor mobile robot to move autonomously to the goal position. A key idea of this localization system consists of fusing the Radio Frequency Identification (RFID) and the vision sensor. [18] presents a simple artificial landmark model and a robust tracking algorithm for the navigation of indoor mobile robots. [19] presents a landmark-based navigation technique for a mobile robot, where the position estimation is achieved by using a camera and a landmark, which consists of simple geometric patterns. [12] proposes a new system for vision-based mobile robot navigation in an unmodeled environment. Simple, unobtrusive artificial landmarks are used as navigation and localization aids. [20] proposes a method based on landmarks able to converge to a position estimate with greater accuracy using less measurements than other frequently used methods, such as Kalman filters. One of the fundamental problem is to integrate a global and local path planning and localization for an automatic recharging system.

3 VineRecharge System

Vinecharger is an automatic recharging system for the AGROB[1] robots generations. Vinecharger has three main components (Fig. 2): Vision Based docking system, Go to the nearest docking station, and Power Monitor.

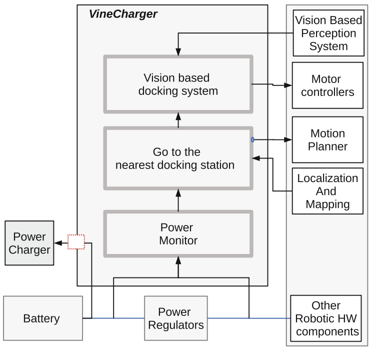
Vinecharger flows according Vinecharger State Machine, which has five states: Idle, Go to Station, Alarm, Docking and Charging. The transition between these states is trigged by the available energy and according a pre-processed charging trajectory and energy map (PCTEM), Fig. 3.

3.1 Off Line Path Planning

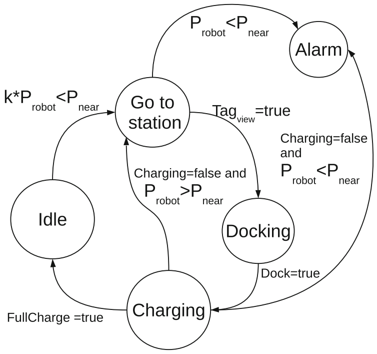
To process the charging trajectory and energy map (PCTEM) represented in Fig. 3, we resorted to an A* algorithm developed in [21] that restricts the path to the maximum turning rate allowed by the robot, assuring that the generated path is feasible for the robot. This A* uses a 16 layered occupation grid map, each layer corresponding to a different orientation, spaced by 22.5°.

Each charging location is located in the center, (a, b), of a parametric circumference (x, y) with a r radius and a t parameter represented in Eq. 1. In Fig. 3,

[1] agrob.inesctec.pt.


VineCharger
Vision Based Perception System
Vision based docking system
Motor controllers
Motion Planner
Go to the nearest docking station
Localization And Mapping
Power Charger
Power Monitor
Battery
Power Regulators
Other Robotic HW components

(a) Hardware architecture



(b) Vinecharger State Machine

Fig. 2. Vinecharger architecture and state machine

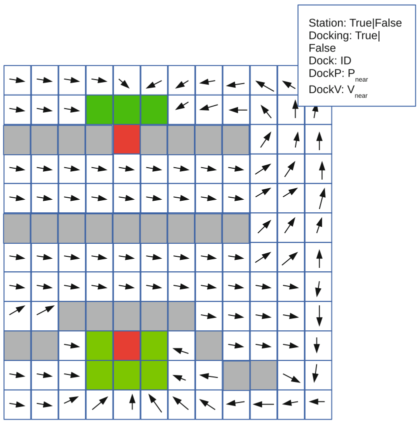


Fig. 3. Preprocessed charging trajectory and energy map.

these charging locations are represented with red cells on the occupation grid map.

$$\begin{cases} x &= a + r\cos(t),\ 0 \leq t \leq 2\pi \\ y &= b + r\sin(t),\ 0 \leq t \leq 2\pi \end{cases} \tag{1}$$

So, the off-line planning method is composed by these three steps:

1. The off line path planning for each cell and each layer:
   (a) Plan a path with origin in the actual cell, to a point (x, y) of the circle that delimits the charging zone, considering $r = 0$. If the A* doesn't find a way, we increase the value of r, and try to plan a new path.
   (b) Estimate the **energetic cost** for the paths generated to every charging station, and save the one with minimum cost.

(c) Go trough every cell occupied by the chosen path, saving that trajectory in each cell so that we don't need to generate a path on every single cell of the map.

3.2 Energetic Cost

The energy consumption between the point $i-1$ and i is given by the Eq. 2, where instead of using the absolute value of the gravitational acceleration, we calculated it's distribution by the three axis in the world (x, y, z), in a such way that the consumed energy increases when the robot is going up the vineyard, and decreases when the robot it's going down.

$$E_{i-i,i} = m \sum_{j=1}^{j_f} (\mu g_{z_j} + a_j) v_j \Delta_t - m \sum_{j=1}^{j_f} (\mu g_{x_j} + a_j) v_j \Delta_t \tag{2}$$

- $m =$ robot's mass
- μ is the friction coefficient, and $g = (g_x, g_y, gz)$ is the gravitational acceleration
- a is the robot's acceleration and v it's his speed
- Δ_t is a small time variation chosen for discretization
- $j \in \mathbb{N}\ [0\ j_f]$, when $u_{j_f} \geq 1$ (u_{j_f} defined in Eq. 5)

The generated path is described by parametric Bézier curves, defined in the Eqs. 3 and 4. The first equation (3) is a linear curve that it's used to describe the trajectory between two waypoints with the same orientation. The second equation (4) is used to describe the trajectory when there's an orientation change.

So, the velocity and acceleration of the robot can be derived from the Bézier curves that describe the trajectory.

$$\begin{cases} x(u) &= (1-u)x_{i-1} + ux_i \\ y(u) &= (1-u)y_{i-1} + uy_i \end{cases} \tag{3}$$

$$\begin{cases} x(u) &= ((1-u)^2)x_{i-1} + 2u(1-u)x_a + u^2 x_i \\ y(u) &= ((1-u)^2)y_{i-1} + 2u(1-u)y_a + u^2 y_i \end{cases} \tag{4}$$

- (x_{i-1}, y_{i-1}) and (x_i, y_i) represent two waypoints
- (x_a, y_a) is a medium point between the two waypoints, that depends on the orientation of each one
- u is a parameter that goes from zero to one, and we calculated it so that the robot speed could be constant (Eq. 5)

$$u_{j+1} = u_j + \sqrt{\frac{k^2}{\left(\frac{\mathrm{d}x}{\mathrm{d}u}\right)^2 |_{u=u_j} + \left(\frac{\mathrm{d}y}{\mathrm{d}u}\right)^2 |_{u=u_j}}} \Delta t \tag{5}$$

3.3 Docking Controller (Bézier Curve)

When the robot reaches near to docking station, eventually a visual tag (placed on docking station) will be detected by the Vision Based docking system. When the visual tag is detected the Vinecharger makes the transition from *Gotothestation* to *Docking* state. The tag relative position and orientation is obtained and then the robot's trajectory is calculated for the docking approach. For that, a cubic Bézier Curve was used in which the starting point is the position of the robot at the moment when it saw the tag and the end point is the position of the tag at that every moment. The points P_i, P_a, P_b and P_f of the Bézier Curve (B_P) are given by the Eq. 6. These points are represented on the right side of Fig. 4.

$$B_P = \begin{cases} x_i = y_i = y_a = 0 & \\ x_a = x_f \times 0.30 & \\ x_f = z_{tag\_detector} & \\ y_f = -x_{tag\_detector} & \\ m_f = \frac{(y_f + sin(pitch - \pi) * 5) - y_f}{(x_f + cos(pitch - \pi) * 5) - x_f} & \\ ma = 0, & \text{if } m_f < 0 \ \& \ y_f < 0 \\ & \text{if } m_f > 0 \ \& \ y_f > 0 \\ ma = \frac{-1}{m_f}, & \text{if } m_f < 0 \ \& \ y_f > 0 \\ & \text{if } m_f > 0 \ \& \ y_f < 0 \\ x_b = x_{target}, & \text{if } m_{target} = \text{inf} \\ y_b = m_a \times x_b - m_a \times x_a + y_a, & \text{if } m_f = \text{inf} \\ x_b = \frac{y_b - y_a}{m_a} + x_a, & \text{if } m_f = 0 \\ x_b = \frac{y_b \times y_f}{m_f} + x_f, & \text{if } m_a = 0 \\ y_b = y_a, & \text{if } m_a = 0 \\ x_b = \frac{x_a \times m_a - y_a - x_f \times m_f + y_f}{m_a - m_f}, & \text{if } m_a \neq \text{inf} \ \& \ m_f \neq \text{inf} \\ & \quad \& \ m_a \times m_f \neq 0 \\ y_b = m_f \times x_b - m_f \times x_f + y_f, & \text{if } m_f \neq \text{inf} \ \& m_a \neq 0 \end{cases} \tag{6}$$

Algorithm 1 presents the pseudocode with the steps of the docking process.

Algorithm 1. Docking process

```
1: Get cubic Bézier Curve points.
2: Transition: global position → relative position.
3: Use a local controller for docking approach.
```

4 Tests and Results

In order to select the most robust and efficient visual tag and detector, in Sect. 4.1 we make an intensive analysis of different visual tags and detector, which will allow us to select one for integration into the vision based docking system block.

Table 1. Results obtained from tests performed using several tags at various distances.

		tag					
dist (m)	arv_dist (m) dist_std_dev (m) avr_time (s)	visp[2]	ar_sys (aruco)[3]	ar_track[4]	april_tags Original[5]	april_tags RIVeR[6]	april_tags Xenobot[7]
0.22	arv_dist	0.263	0.209	0.206	0.218	0.207	0.216
	dist_std_dev	0.0012486	0.0008511	0.0014431	0.0005408	0.0000964	0.0012097
	avr_time	———	———	———	0.02585	0.15160	0.00407
0.40	arv_dist	0.465	0.377	0.375	0.398	0.376	0.378
	dist_std_dev	0.0003055	0.0000947	0.0001175	0.0000951	0.0000628	0.0001902
	avr_time	———	———	———	0.02555	0.16963	0.00427
0.60	arv_dist	0.707	0.566	0.558	0.603	0.558	0.570
	dist_std_dev	0.0005070	0.0000889	0.0002518	0.0001393	0.0000776	0.0006230
	avr_time	———	———	———	0.02483	0.16531	0.00478
1.00	arv_dist	1.181	0.955	0.947	1.012	0.935	0.956
	dist_std_dev	0.0030589	0.0003523	0.0011032	0.0003705	0.0004184	0.0005425
	avr_time	———	———	———	0.02708	0.18139	0.00510
1.20	arv_dist	1.406	1.140	1.126	1.215	1.125	1.155
	dist_std_dev	0.0066985	0.0009085	0.0016255	0.0003829	0.0003926	0.0056493
	avr_time	———	———	———	0.02943	0.19018	0.00532
1.50	arv_dist	1.757	1.430	1.413	1.521	1.401	1.457
	dist_std_dev	0.0066303	0.0046561	0.0010171	0.0015644	0.0004264	0.0027019
	avr_time	———	———	———	0.03046	0.19906	0.00548
1.70	arv_dist	1.999	1.630	1.610	1.705	1.594	1.623
	dist_std_dev	0.0211482	0.0011948	0.0011958	0.0012763	0.0014883	0.0087405
	avr_time	———	———	———	0.03278	0.19834	0.00532
2.00	arv_dist	2.347	1.917	1.889	2.005	1.871	1.927
	dist_std_dev	0.0155401	0.0005048	0.0042226	0.0020556	0.0011278	0.0050850
	avr_time	———	———	———	0.03183	0.21361	0.00534

4.1 Visual Tags Detector Benchmarking

Six visual tags and detectors were tested in terms of accuracy, robustness, and processing time of various solutions available for detection of artificial landmarks.

The table 1 summarizes the results of the tests performed using four types of tags, however six implementations were tested: visp,[2] ar_sys,[3] ar_track,[4] april_tags original,[5] april_tags RIVer,[6] april_tags Xenobot.[7] This benchmarking was performed in an indoor environment and the HD Web Camera (1280×720 resolution) of the ASUS K750JB notebook with an Intel Quad Core i7-4700HQ (@2.4 GHz) processor, 12 GB of RAM and Ubuntu 14.04 LTS was used.

For each one tested distance was obtained 1000 measurements (*avr_dist*), the standard deviation of these measurements (*dist_std_dev*) as well as the average

[2] VispAutoTracker implementation: http://wiki.ros.org/visp_auto_tracker.
[3] ArSys implementation: http://wiki.ros.org/ar_sys.
[4] ArTrackAlvar implementation: http://wiki.ros.org/ar_track_alvar.
[5] AprilTags original implementation: http://people.csail.mit.edu/kaess/apriltags.
[6] AprilTags RIVeR-Lab implementation: http://wiki.ros.org/apriltags_ros.
[7] AprilTags Xenobot implementation: https://github.com/xenobot-dev/apriltags_ros.

Table 2. Characteristics of the detection algorithms.

visp	min_dist: ~18 cm max_dist: ~400 cm	• Very sensitive to variations in brightness; • To perform detection, the tag has to be viewed closely (< 40 cm) (a tag that appears far from the camera is not detected); • Provides distance values that are farthest from real values.
ar_sys (aruco)	min_dist: ~22 cm max_dist: ~235 cm	• Sensitive to variations in brightness; • It has a lower maximum range.
ar_track	min_dist: ~17 cm max_dist: ~350 cm	• It confuses the tags id (sometimes the algorithm detects a new tag when in fact it is the same).
april_tags Original	min_dist: ~17 cm max_dist: ~460 cm	• More accurate distance values; • Low processing times.
april_tags RIVeR	min_dist: ~17 cm max_dist: ~460 cm	• Relatively accurate distance values; • It does not work very well at short distances (less than 40 cm fails some detections); • Very high detection times.
april_tags Xenobot	min_dist: ~20 cm max_dist: ~460 cm	• Relatively accurate distance values; • Very low detection times

tag detection time (*avr_time*). The value of the *dist* is given by the tested algorithms and the *time* corresponds to the elapsed time since the detection of the tag until its distance and position are provided. For the *visp*, *ar_sys* and *ar_track*, the *time* parameter could not be measured. This procedure was repeated for eight different distances between 22 cm and 200 cm.

Table 2 shows some characteristics of the detection algorithms that were identified during the tests.

As it is possible to verify through the Tables 1 and 2, the best solution is the original implementation of *AprilTags*, because it was the one that provided distance values closer to the real and has a very low standard deviation of the measurements. However, this implementation does not provide a ROS node that publishes the information (position and orientation) of the identified tag so that it can be used by other nodes. So two implementations have been tested that publish this information. Of these two implementations the best was the *AprilTags Xenobot* implementation because, compared to the *AprilTags RIVeR-Lab* implementation, the distance values (although a little distant) are closer to the real values and the average processing time is much shorter. In this way the *AprilTags Xenobot* implementation was chosen for the further development of the system, not because it stands out from the other implementations but because it does not present the problems of the same ones.

4.2 Docking and Off Line Path Planning Test

In Fig. 4, on the left side an example of the detection of a tag is shown and on the right side is shown the respective trajectory that the robot must perform to correctly connect to the docking station. This trajectory is estimated by the Docking Controller, Sect. 3.3.

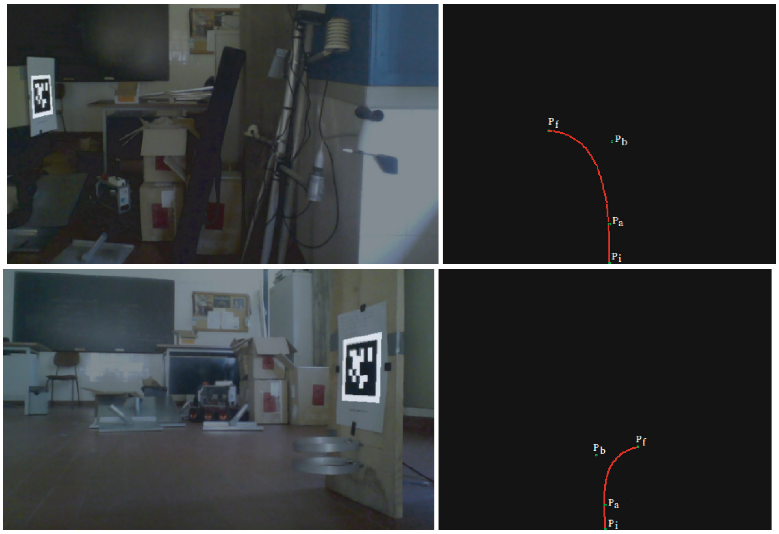


Fig. 4. Detected tag (left) and Bezier Curve obtained (right).

4.3 Tests and results for energy consumption

Considering the Agrob simulation framework [5] and the next parameters for the Energetic Cost algorithm: $m = 1\ kg$ $\mu = 0.06$ $g = 9.8\ ms^{-2}$ $v = 1\ ms^{-1}$ $a = 0\ ms^{-2}$ $\Delta t = 0.001s$ $C_{mass} = (0, 0, 0.4)m$. Two paths with different lengths were tested, one going down and other going up, with the results shown in the Table 3. As expected, the energy consumption increases with distance and when going uphill.

Graphically, Fig. 5 represents a graphic with the gravitational force imposed by the terrain to the robot, $g = (g_x, g_y, g_z)$, and the energy consumption expressed in Joules, during a downhill from altitude 12 m to 0 m and a distance of 10 m. In Fig. 5, we can observe that in a downhill path the gravitational force in g_x takes positive values, meaning that it's a force that helps to the robot movement, being visible an attenuation in the energy consumption. The opposite

Table 3. Energetic consumption

Energy (J)	19 m		9 m	
	Downhill	Uphill	Downhill	Uphill
1	8.5	12	3.7	4.7
2	8.4	11	3.6	4.9
3	8.48	10.95	3.3	4.8
Mean	8.46	11.32	3.53	4.8

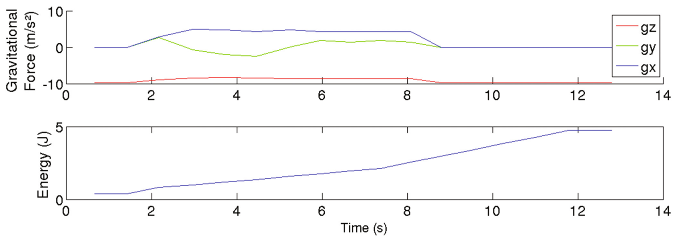


Fig. 5. Graphical representation of gravitational acceleration and consumed energy during downhill.

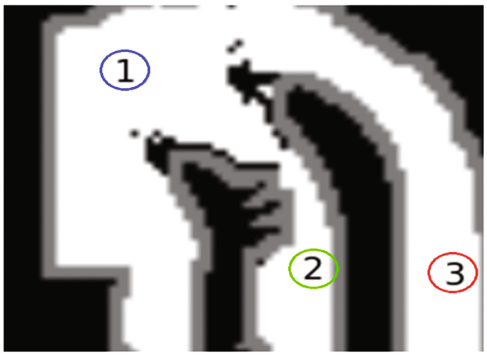


Fig. 6. Location of the charging stations

would happen during an uphill, where g_x would take negative values, dragging back the robot, and consequently, the energy consumption would increase.

4.4 Tests and Results Off-Line PCTEM Processing

To test off-line PCTEM processing, we have considered the occupation map from AGROB simulation framework [5] and three charging stations, Fig. 6, identified by numbers: 1, 2 or 3, and color: blue, green or red, respectively. Each station is located in a different altitude, being the station number 1 the one with minimum altitude, and the number 3 the station with maximum altitude.

The final results, for four possible layers of the occupation grid map, is represented in Fig. 7, where a blue cell means that the chosen station for the recharge was the station number 1. A green cell shows that station 2 was the more adequate, and a red cell represents the choice of station number 3. It's observable, a predominance of the blue color, meaning that the station with the minimum altitude is the most requested one, as their paths require less energy consumption. To process the PCTEM the algorithm has taken 90 min (with CPU AMD A10-8700P Radeon R6, 1.3 GHz). However the path planning can be immediately obtained from the PCTEM, by using the directional vector stored in each cell of PCTEM.

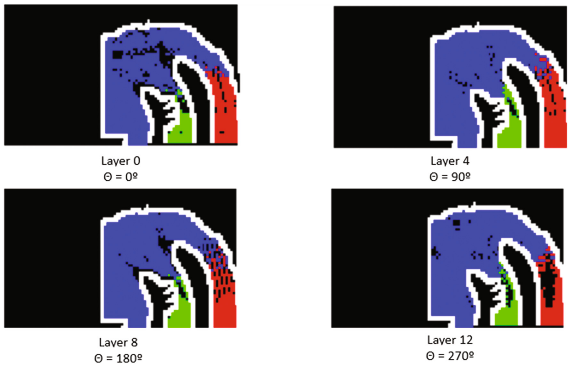


Fig. 7. Results for the off-line planning

5 Conclusion

The proposed VineRecharge system was developed considering the constrains of a cost-effective robot to carry-out crop monitoring tasks in steep slope vineyards, and reduces computational power required by the robot even in situation of obstacle avoidance, for example can be easily integrated with Virtual Force Field (VFF) and the Vector Field Histogram (VFH) Methods. Besides, the path planning to the docking station can be immediately obtained from the PCTEM, by using the directional vector stored in each cell of PCTEM. In terms of visual tags, the *AprilTags Xenobot* implementation was chosen for the further development of the system, not because it stands out from the other implementations but because it does not present the problems of the same ones. As future work we intend to carry out the tests in a real vineyard step slope scenario.

Acknowledgment. This work is financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project "POCI-01-0145-FEDER-006961", and by National Funds through the FCT – Fundaçao para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

References

1. euRobotics: Strategic research agenda for robotics in Europe Information (2013). http://ec.europa.eu/research/industrial_technologies/pdf/robotics-ppp-roadmap_en.pdf
2. Bac, C.W., Henten, E.J., Hemming, J., Edan, Y.: "Harvesting robots for high-value crops: state" of-the-art review and challenges ahead. J. Field Robot. **31**(6), 888–911 (2014)
3. VineRobot - FP7 project. Information. http://www.vinerobot.eu/
4. Vinbot - FP7 project. Information. http://vinbot.eu/?lang=pt

5. Dos Santos, F.N., et al.: Towards a reliable robot for steep slope vineyards monitoring. J. Intell. Robot. Syst. Article 340 (2016). https://doi.org/10.1007/s10846-016-0340-5
6. Won, P., Biglarbegian, M., Melek, W.: Development of an effective docking system for modular mobile self-reconfigurable robots using extended kalman filter and particle filter. Robotics **4**(1), 25–49 (2015)
7. Kim, M., Kim, H.W., Chong, N.Y.: Automated robot docking using direction sensing RFID. In: 2007 IEEE International Conference on Robotics and Automation, pp. 4588-4593 (2007)
8. Mintchev, S., et al.: Towards docking for small scale underwater robots. Auton. Robots **38**(3), 283–299 (2015)
9. Zhu, Y., et al.: A multi-sensory autonomous docking approach for a self-reconfigurable robot without mechanical guidance. Int. J. Adv. Robot. Syst. **11**, 10 (2014)
10. Roh, S., Park, J.H., Lee, Y.H., Song, Y.K., Yang, K.W., Choi, M., Kim, H.-S., Lee, H., Choi, H.R.: Flexible docking mechanism with error-compensation capability for auto recharging system of mobile robot. Int. J. Control Autom. Syst. **6**(5), 731–739 (2008)
11. Acuna, R., Li, Z., Willert, V.: MOMA: Visual Mobile Marker Odometry (2017). arXiv preprint arXiv:1704.02222
12. Briggs, A.J., Scharstein, D., Braziunas, D., Dima, C., Wall, P.: Mobile robot navigation using self-similar landmarks. In: 2000 Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2000, vol. 2, pp. 1428-1434 (2000)
13. Dutta, V.: Mobile robot applied to QR landmark localization based on the keystone effect. In: Mechatronics and Robotics Engineering for Advanced and Intelligent Manufacturing, pp. 45–60 (2017)
14. Yoon, K.J., Jang, G.J., Kim, S.H., Kweon, I.S.: Fast landmark tracking and localization algorithm for the mobile robot self-localization. In: IFAC Workshop on Mobile Robot Technology, pp. 190–195 (2001)
15. Lo, D., Mendonça, P.R., Hopper, A.: TRIP: A low-cost vision-based location system for ubiquitous computing. Pers. Ubiquit. Comput. **6**(3), 206–219 (2002)
16. Zeybek, T., Chang, H.: Delay tolerant network for autonomous robotic vehicle charging and hazard detection. In: 2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS), pp. 79–85 (2014)
17. Chae, H., Han, K.: Combination of RFID and vision for mobile robot localization. In: Proceedings of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing Conference, pp. 75–80 (2005)
18. Jang, G., Lee, S., Kweon, I.: Color landmark based self-localization for indoor mobile robots. In: 2002 Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2002, vol. 1, pp. 1037–1042 (2002)
19. Lin, C.C., Tummala, R.L.: Mobile robot navigation using artificial landmarks. J. Field Robot. **14**(2), 93–106 (1997)
20. Boley, D.L., Steinmetz, E.S., Sutherland, K.T.: Robot localization from landmarks using recursive total least squares. In: 1996 Proceedings of the IEEE International Conference on Robotics and Automation, vol. 2, pp. 1381-1386 (1996)
21. Fernandes, E., Costa, P., Lima, J., Veiga, G.: Towards an orientation enhanced astar algorithm for robotic navigation. In: 2015 IEEE International Conference on Industrial Technology (ICIT), pp. 3320–3325 (2015)