Programming a Thorvald robot to count grape bunches in a vineyard

UNIVERSITY OF LINCOLN

line 1: 2nd Stephen Rerri-Bekibele
line 2: *School of Computer Science. University of Lincoln (of Affiliation)*
line 3: Lincoln, United Kingdom
line 4: 16663359@students.lincoln.ac.uk

***Abstract*—Accurately counting fruit in a farm has always been a job only human beings could do. Now with emerging technologies robots can do this much more accurately and faster than human beings can. The aim of this project is to demonstrate this capability using a Thorvald robot in a physics simulation called gazebo [1] in conjunction with RVIZ. This report documents the findings of using the Thorvald robot to count the number of grape bunches in a variety of vineyard structures. As this model of robot has 3 cameras, all of them were used to increase the efficiency of the counting while reducing the time taken. The algorithm allows the robot to work with minimal user input in a variety of environments and for different fruits. This is possible due to the use of 3 core python scripts. The result demonstrated that the robot can accurately count the number of a given grape bunch in a test environment. However, it needs ample time to count the grapes and this time is provided by using a wall following algorithm to make the robot loop around grape vines in a circle until counting is done. The number of times this looping occurs can be hard coded or simply moving the robot to another location for it to continue a count is also possible at runtime. Along with the wall following algorithm, a grape searching algorithm was also created. This allows the robot to actively seek out and move to grape bunches. These two algorithms combined using a state machine would make a highly autonomous and efficient fruit counting robot. Unfortunately, the current state of the project has the two algorithms working separately as when they are launched without a state-machine, conflicting parameters result in the robot getting stuck at times around corners and moving oddly at others.**

Keywords—counting, maximize, model, cameras, simulation (*key words****)***

INTRODUCTION

At the University of Lincoln, the Thorvald robot, supplied through a partnership with SAGA robotics is the base model robot spearheading scientific research throughout farms for the purpose of moving towards a robotics centred farming future. Through applying ROS techniques to simulate how the real Thorvald robot would be capable of counting fruit bunches in the real world and by using clever computer vision techniques

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

and effective positioning, the robot is able to autonomously provide an accurate count of any fruit the cameras are tuned to detect. One key objective of this project was to stand out from the rest and make effective use of all the features of the robot that were available, namely using front, right and left cameras so the robot could count regardless of which way it was facing. This project is significant because it allows the robot to have a high level of autonomy due to its obstacle avoidance and ability to actively detect, seek out grape clusters and position the robot next to them but also because of the use of 3 cameras, enabling the robot to count multiple rows when active. This is NOT done in parallel but rather sequentially by allowing each camera to access the counting algorithm with a library of message filtering algorithms called message filters [2].

RELATED WORK

Fruit detection with mobile robots

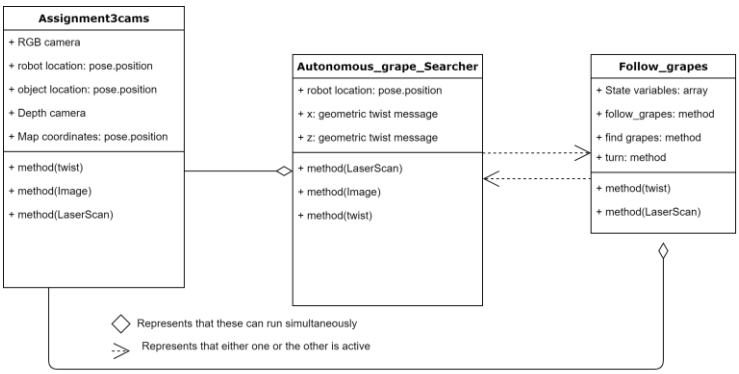
Robots come in all shapes and sizes, and this means they come with all sorts of sensors and actuators. But the one thing that remains the same in simulation is that robots all have a way of interacting with objects around them. There are a variety of ways robots can be used to map fruits in simulated environments. One approach is demonstrated in "Fruit mapping mobile robot on simulated agricultural area in Gazebo simulator using simultaneous localization and mapping (SLAM)" (N. Habibie et al.) [3]. Which makes use of SLAM "by generating grid-based/volumetric map using fine-tuned SLAM-Gmapping algorithm and combine it with properties/ information obtained from each detected crops/plants using fruit detection with visual sensor and tree location detection using 2D laser scanner sensor". This research paper achieves a similar objective of counting fruit bunches but utilises a different approach, a Husky Robot equipped with a "simulated sensor of 2d Laser Range sensor LMS1 and one visual sensor Kinect (only using 2D image, depth data is not used)" [3] the benefit of which is that it does not require the use of a depth camera and yet still boasts a good result with good accuracy. This accuracy comes from the "generated 2D volumetric map of agricultural area which is enriched with information/properties for each tree". What makes SLAM unique is "the process of mapping and localization is done concurrently and recursively" [4].

The problem with this approach is the approach itself. To make a good map of a robot's environment the robot needs a precise self-position estimation, however good localization can only be achieved when a well-defined map is available. Therefore, SLAM or "Concurrent Mapping and Localisation" is a term known as an approach to solve a "chicken-and-egg problem" of robot localization and mapping. In addition, this approach was made even more difficult because SLAM is designed to work well in indoor environments with confined spaces, as this is not the same outdoors, the SLAM-Gmapping had to be adjusted to become more suitable with outdoor environments.

Another approach which was motivated by the inability for RGB cameras, used for detection, to work in variable lighting was the use of a Mobile terrestrial laser scanner (MTLS), composed of a Velodyne VLP-16 LiDAR synchronised with RTK-GNSS satellite navigation receiver to detect and localise Fuji apples [5]. The main advantage of this approach being that it can work at night and with no light which is a significant advantage over the aforementioned approach and the approach this paper covers. Nonetheless, this advantage is a double-edged sword with a steep disadvantage and that is because of the use of a satellite, which means that the system would not work indoors as well as in areas with thick vegetation or where there is obstruction between the robot and the satellite, such as on cloudy days or in urban areas. This does not deter from the fact that the use of the lidar and satellite, while constrained, still achieve a detection rate similar to those "obtained by RGB-based systems, and with the additional advantages of providing direct 3D fruit location information" [5].

CONCEPT

The conceptual model that defines the architecture of my system is very simple and can be best described as a state machine [6] combined with "persistently active script" that waits for incoming data from the simulation topics it is communicating with.



1. SOFTWARE DIAGRAM OF THE CONNECTION BETWEEN THE DIFFERENT SCRIPTS IN THE PROJECT.

The persistently active script was comprised of A script "Assigmnent3cams", that takes in data from the RGB and depth camera topics from all three cameras and then using a "utility library, message filters, which collects commonly used message filtering algorithms into a common space" [7], the data from these cameras are sequentially parsed into a function "image_cb" which uses the data to create a X,Y,Z coordinates for a 3D Point for all objects the cameras pick up. To count the grapes the cameras were tuned using the OpenCV library, home to a collection of algorithms for image processing, [8] with an algorithm "inRange" that takes in a user designed set of BGR values of interest. These values then get converted to HSV using a second OpenCV algorithm that is necessary for the cameras to filter out unnecessary data by converting into a black and white format for the next stage of image processing. This next stage involved yet another OpenCV algorithm called "Morphological Transformation". This enabled the final image data used to detect contours to be as accurate as possible with techniques that "filled in small holes inside the foreground objects as well as removing small black points" [8] in the image stream, the result of 'noise' data that un-intentionally got through the OpenCV "inRange" algorithm.

As this Project was done in a physics simulation, this meant the values for the points were constantly changing with a variation of + or – 2 decimal places, to resolve this, a density-based algorithm used for clustering spatial-temporal data was used. This algorithm is called DBSCAN, and it was chosen due to its ability to "find core samples of high density and expands clusters from them", making it optimal "for data which contains clusters of similar density" [9] As was the case in the physics simulation with each object having a constantly varying set of coordinates.

I choose to use a state machine because "It is a general method for implementing fault-tolerant services" (Schneider, F.B., 1990.) [6] This enabled my wall following algorithm which operated on the state machine approach to efficiently switch between the states I had defined, consistently and most importantly, autonomously. This approach was dissimilar to those of others carrying out a similar object of getting the robot to and moving around an object by providing "as output a sequence of waypoints to be followed" (Gutierrez, R et al) [10].

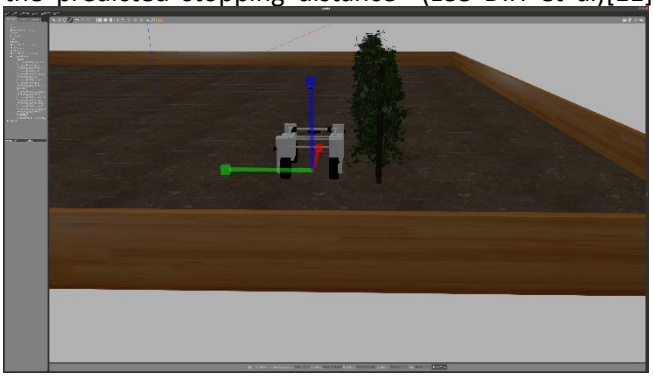
The advantage of which are that the robot will never deviate from the waypoints and so will keep on path unless the waypoints are moved or unavailable.

Secondly, Waypoint-centric functionality such as "a Modular and scalable waypoint tracking controller" (Gutierrez, R et al) [10] Can enable the robots to move at high speeds in urban environments (up to 50km/h) by performing a smooth interpolation of the waypoints and using optimal control techniques to ensure robust trajectory tracking. The demerit of this is that a control loop is needed to stabilize the system due to the "delays in the localization system and actuators" (Gutierrez, R et al) [10].

The advantage of my system over this is that no waypoints are required to direct the robot and as the robot operates on a closed feedback loop to ensure it is moving as instructed, it does not require a localization system. The robot will simply not deviate so long as the environment does not change. Should the robot be required to operate in a different environment it can be programmed to do so by defining new states that define how it should interact with its new surroundings.

1. Image of positioning the robot next to a "grape wall" so it can begin counting

The main demerit of my approach is that it is limited by the range of the sensors which determine how and when the robot should turn or slow down. At high speeds this could be a problem simply because of stopping distance and breaking distance. In order to operate at high speed my robot would require "An autonomous braking control algorithm designed using the predicted stopping distance" (Lee D.H et al)[11]

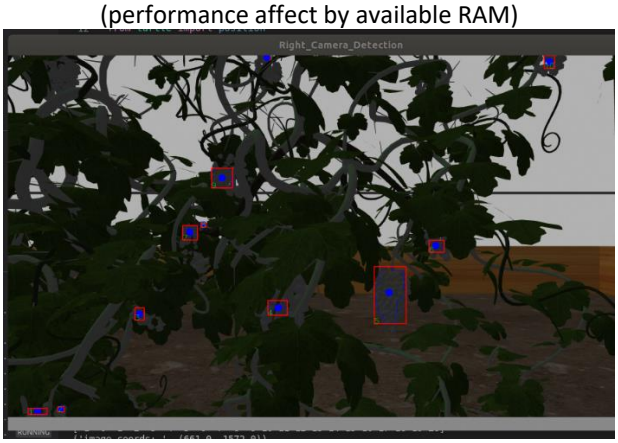


2. Additional angle of robot next to "grape wall"

EVALUATION

The Robot performed with a counting accuray of 90%. This was because although the robot could detect the grapes with 100% accuracy as seen below, the parameters for the algorithm used to determine if a grape had been seen before were not appropriately tuned.

3. Shows how the camera could detect grapes in view (performance affect by available RAM)



4. Right camera detection of grapes

5. Accurate mask of the detected grapes, generated from the OpenCV in range and OpenCv's Morphological transformation algorithm and used to navigate to the grapes or count.

The requirements were set too high and therefore the robot initially may not count a grape bunch in one loop of the vineyard because it believed it had come across it before. This mean the number of times the robot had to come across the same bunch of grape (it hadn't counted) for it to finally be counted increased. Therefore the simulation time also increased. This demonstrated that given enough time the robot would eventually count all grapes to a higher degree of accuracy. The Algorithm used to determine if a grape bunch had been detected before is called DBSCAN, as previously mentioned it is part of the Sklearn.clustering library [9].

Furthermore, the algorithm was not tuned for speed with Python, the programming language used for the simulation and therefore the confirmation rate could've been even faster.

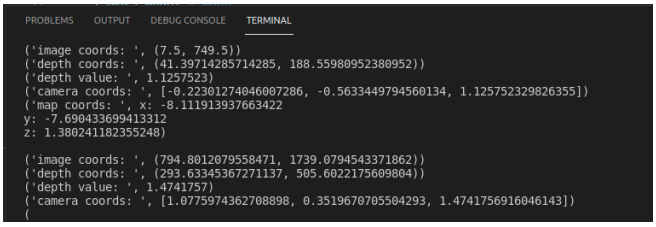
At the current state of the project the values for the clustering algorithm were as follows:

Figures and Tables

A. DBSCAN values used

Eps	0.05 "Decimal places"
Min_samples	17 "Integers"

This meant that each new grape bunch could have a map coordinate value that was only similar to that of any persisting detected ones by + or - 0.05 as defined by eps and there could be as many as 17 grape bunches treated as the same grape inclusively. This ensured the high level of accuracy but also meant that some grapes were skipped if there too close physically to other grapes. Especially as this was done in a simulation, some grape bunches merged into other grape bunches and were then counted as 1 when they should have been.



6. Example of map coordinate values

Notes for improvement on this project include perfecting the state machine so the robot does not have to be placed next to the grape wall to begin detecting, this can be resolved by properly combining the actions of the state machine script Follow_grapes, that handles carefully directing the robot around grape

walls for counting, to the Autonomous grape_searcher script that is used to detect grapes and drive the robot to them, including built in obstacle avoidance and autonomous searching when no grapes are in sight. I would also like to implement some additional features to output the time taken for all grapes to be detected and counted and then output the data to a text file for summary statistics with MATLAB [12] or another capable tool. Lastly, I would like to run the simulation without a GUI for faster performance and data retrieval to determine the best values for the DBSCAN algorithm to have the most accurate counting and for the grapes to be counted in the shortest amount of time.

7. TABLE OF RESULTS FROM MULTILE RUNS OF THE SIMULATION

Table Head	**Number of Iterations**		
	Grapes detected (integers)	***Time taken (Minutes)***	***NO. of vineyard loops (Integers)***
1	75	10	20
2	80	15	25
3	20	1.5	4
4	60	5	15

ACKNOWLEDGMENT *(Heading 5)*

I would like to acknowledge the Following Lecturers and colleagues at the University of Lincoln for their Brilliant Teaching and for bringing to light, techniques that were applied to this project:

Prof. Marc Hanheide

Dr Gregorz Cielniak

Nikolaos C Tsagkopoulos – for showing me how to use message filters

REFERENCES

[1] Koenig, N. and Howard, A., 2004, September. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)* (Vol. 3, pp. 2149-2154). IEEE.

[2] Anwar, K., Wibowo, I.K., Dewantara, B.S.B., Bachtiar, M.M. and Haq, M.A., 2021, September. ROS Based Multi-Data Sensors Synchronization for Robot Soccer ERSOW. In *2021 International Electronics Symposium (IES)* (pp. 167-172). IEEE.

[3] Habibie, N., Nugraha, A.M., Anshori, A.Z., Ma'sum, M.A. and Jatmiko, W., 2017, December. Fruit mapping mobile robot on simulated agricultural area in Gazebo simulator using simultaneous localization and mapping (SLAM). In *2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS)* (pp. 1-7). IEEE.

[4] Khairuddin, A.R., Talib, M.S. and Haron, H., 2015, November. Review on simultaneous localization and mapping (SLAM). In *2015 IEEE international conference on control system, computing and engineering (ICCSCE)* (pp. 85-90). IEEE.

[5] Gené-Mola, J., Gregorio, E., Guevara, J., Auat, F., Sanz-Cortiella, R., Escolà, A., Llorens, J., Morros, J.R., Ruiz-Hidalgo, J., Vilaplana, V. and Rosell-Polo, J.R., 2019. Fruit detection in an apple orchard using a mobile terrestrial laser scanner. *Biosystems engineering*, *187*, pp.171-184.

[6] Schneider, F.B., 1990. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, *22*(4), pp.299-319.

[7] J. Faust, V. Pradeep, D. Thomas. Messaged Filters. Available at: http://wiki.ros.org/message_filters

[8] Bradski, G., 2000. The openCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, *25*(11), pp.120-123.

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay, Scikit-learn: Machine Learning in Python; 12(85):2825−2830, 2011.[Available at: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html]

[10] Gutiérrez, R., López-Guillén, E., Bergasa, L.M., Barea, R., Pérez, O., Gómez-Huélamo, C., Árango, F., Del Egido, J. and López-Fernández, J., 2020. A waypoint tracking controller for autonomous road vehicles using ROS framework. *Sensors*, *20*(14), p.4062.

[11] Lee, D.H., Kim, S.K., Kim, C.S. and Huh, K.S., 2014. Development of an autonomous braking system using the predicted stopping distance. *International Journal of Automotive Technology*, *15*(2), pp.341-346.

[12] Etter, D.M., Kuncicky, D.C. and Hull, D.W., 2002. *Introduction to MATLAB*. Hoboken, NJ, USA: Prentice Hall.