POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

An EKF-SLAM approach to plant counting

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Erica Ceriotti

Advisor: Prof. Matteo Matteucci

Co-advisors: Paolo Cudrano, Simone Mentasti

Academic year: 2022-2023

1. Introduction

According to predictions, the global population will surpass 9 billions people by 2050. The food, and especially, the crop production needs to be increased to satisfy the demand, but this comes at a cost for the environment. Indeed, one of the most used types of fertilizers, the Nitrogen, becomes dangerous when dispersed. Precision agriculture is a strategy which consists in the collection and processing of data from the fields in order to support management decisions. It has been created to face the aforementioned challenges and finds its application especially in agricultural robotics, a field which has seen its development in the last years due to the lack of human workforce. Various platforms have been created for performing plant monitoring and other types of agricultural activities such as weeding, seeding, harvesting, spraying. Nowadays, the first mentioned task is usually executed manually, with high costs in terms of salaries and time, especially for big realities where the necessity to assess the plants status has to be repeated several times. For example, the orchard nurseries' income depends on their capability of satisfy the customers requests, which is related to the up-to-date estimate of the amount of stock available to be sold. Indeed, small plants can perish due to natural factors and this has to be taken into account in the decision making process. This thesis presents an approach to the automation of the plant counting task with the EKF-SLAM algorithm, realized through the Robot Operating System (ROS), to create a map of the field, from which later assessing the number of plants, parallel to the estimation and correction of the robot trajectory.

2. State of the art

Different solutions have been proposed during the years to address the task at hand, which have been researched and divided depending on the type of sensor employed.

2.1. Infrared sensing

In 2011, the use of an InfraRed (IR) Sensor has been experimented [2]. The results showed that the system overestimated the manual count of less than 5%. The use of the plant width to perform recognition in the collected data, however, showed to be susceptible to the presence of weeds and plants foliage. Also, the sensor needs to be positioned at a certain distance from crop and at certain height to properly detect plants, a condition which is not always possible to maintain especially in irregular terrains.

2.2. Camera sensing

In [1] consumer-graded RGB camera has been mounted on the side of a robot under the name of "TerraSentia" which was driven through fields of corns at various growth stages for assessing the number of plants. The presented system exploited a Convolutional Neural Network (CNN) to perform the recognition. Since CNNs require computational power to operate, the authors used a network of the family of MobileNets to reduce the load. This project introduced the problem of detecting multiple plants at the same time, which was solved in counting the number of plants inside a Region Of Interest (ROI). A disadvantage of such methodology is that it relies on the assumption of a minimum distance between subsequent plants, which can not always be assured in presence of leaves.
Even if the system demonstrated good capabilities, reaching an high Pearson Correlation index of 0.96, compared to the manual counting, we still have to take into consideration that the Neural Network required the collection of thousands of images for training. Also, the quality of cameras captures is sensitive to changes in light conditions which are typical of an outdoor environment, such as an orchard. Also, the work did not presented any information about the battery duration of the robot, which can be determinant in the applicability of such system especially for big orchards.

2.3. LiDAR sensing

Light Detection and Ranging (LiDAR) sensors, of the category of Time-Of-Flight (TOF), emits beams of light in the environment and detect the distance of objects based on the time that it takes for the light to scatter back to the sensor. The result of this process comes in the form of a point cloud, i.e. a set of points which contain information about the position of the sensed element. These sensor are available in different types and offer different resolutions depending on the number of dimensions sensed, spanning from 1D to 3D.
The usage of 2D and 3D LiDAR sensors in plant counting has been demonstrated by different works. In particular, [5] described a 3D LiDAR positioned on the front of a robotic platform named BoniRob, coupled with a GPS for location information. Since the ground was also visible by the sensor, the Random Sample Consensus (RANSAC) algorithm, i.e. an algorithm for the segmentation of a geometric models from a point cloud, in this case a plane, was employed to search for and discard those points belonging to the terrain. To perform plant detection, the plant points were individuated by exploiting a region growing algorithm. The so formed groups of points, also called clusters, were filtered based on their probability of being a plant based on a model created for this purpose.
By using a LiDAR, however, the same plant was visible multiple times. To avoid double counting and improve localisation, a tracker was assigned to each new plant detected and re-matched to the same plant based on the distance from it. With this system the detection and localisation performance increased from 60-70% to 80-90%. The main problems in the detection phase were due to the plants not corresponding to the estimated model.
Despite LiDARs are more costly from an economical point of view with respect to cameras, they are not affected so much by changes in the light conditions as them. Considering the fact the main environment for nurseries develops outdoor, this sensor has been chosen for plant counting also in this Thesis.

3. Proposed approach

We present an approach for the recognition and counting of plants starting from the point cloud registered by a 3D LiDAR. Therefore, the proposed system can be used on already existing agricultural machines, thus relieving the economic cost that a full proprietary system would require. Moreover, this comes with the advantage of the possible parallization of the assessment of the number of plants with other in-field activities such as weeding.
Our approach includes mainly tree steps: plant detection, plant mapping and finally plant counting.

3.1. Plant detection

The process begins with the filtering of an area of the point cloud unneeded for the computation and the detection of plants. As it happens in [5], also in our case the ground is visible from the point cloud, therefore to extract its points we modelled the terrain as a plane as well and

we used the RANSAC algorithm to find the coefficients in the equation of such plane:

$$ax + by + c = d \tag{1}$$

RANSAC does this task by exploiting a voting procedure: it selects a number of random points equal to the one needed to estimate the model, three for planes, it fits the plane to them and it counts the total number of points in the cloud which are close enough to the estimated plane, called inliers, according to a given threshold. Every time a new plane is found, the number of inliers is assessed and if it surpasses a minimum number required for the model to be considered good, the error of the model regarding all of them is computed. The model is then saved if the estimated error is less than the error found so far and all this phases are repeated until the maximum number of iterations allowed is exhausted.
The point cloud, however, changes from frame to frame as the robot moves, therefore, we estimate the plane equation at time $t+1$ by fitting the equation found at t to the new point cloud at $t+1$ and by searching for a new equation from the inliers at $t+1$. In this way we add consistency to the plane estimation, since we expect the plane to be reasonably at the same place in the point cloud.
Once the ground points have been deleted from the scene, the number of points is reduced to keep only those near to the robot as the farer ones are explored later. The plants points are located through the use of the Density-Based Spatial Cluster Algorithm (DBSCAN), which gives the advantage with respect to other clustering algorithms of properly defining the notion of outliers as points not belonging to any cluster. This algorithm groups the points by their nearness and density. It scans the whole point cloud in the search of "core points", i.e. points which are surrounded in a certain distance ϵ by a minimum number of other points, from which to start the creation of clusters. The neighborhood is added to the same cluster of the core point and then it is inserted into a queue for later inspection in search of other points to be merged into the cluster. The process of adding and analysing the new points added is repeated for all those points that satisfy the requirement of a minimum number of points within ϵ as it happens for the core point. Whenever this test is not passed, the cluster expansion stops. This feature of DBSCAN makes it more appropriate for situations where the distance between one cluster and the other is not sufficient to tell them apart, as it might happen with clusters for plants that are too near each other, especially in the leaves part.
The so found clusters are inspected and filtered based on the maximum number of points, the maximum width and the minimum height for a cluster to be considered a plant. Such thresholds can be imposed to eliminate small but different clusters, which might originate from the same plant, or parts of the ground which have passed through the ground detection phase. As last part, the baricenter of the remaining clusters expressed in sensor coordinates is computed by averaging the points over the tree dimensions.

3.2. Plant mapping

When a plant is detected, it has to be followed through its movements to avoid double counting. In addition to this problem, in an orchard environment the GPS signal might not be always available because of plant crowns blocking it. In such case, we would need to rely only on the odometry, i.e. a rough estimate of the position computed by sensors, such as rotary encoders, for the robot localisation. The odometry trajectory, however, diverges from the real path traversed over time. To address both of the mentioned issues, we followed a Simultaneous Localisation and Mapping (SLAM) approach to estimate the map of the field, while correcting the trajectory traced by the robot.
In particular, we used the EKF-SLAM algorithm, which registers maps made by landmarks, i.e. points of interest in the space, composed in this case by the (x_m, y_m) position of each plant detected. The backbone of EKF-SLAM is made by the Extended Kalman Filter (EKF), an algorithm for the estimation of unknown state variables, here represented by the position (x_R, y_R) and the orientation θ of the platform, in non-linear dynamic systems. The EKF, starting from noisy information sources, such as the odometry, is able to produce a more accurate estimate for (x_R, y_R, θ_R) by integrating data coming from other sensors, in this case the plants baricenters of the previous step. In the Ex-

tended Kalman Filter, the state is represented by a multivariate Gaussian $\mathcal{N}(\mu_t, \Sigma_t)$, in presence of multiple state variables.

Algorithm 1 EKF Algorithm

1: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
2: $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
3: $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
4: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
5: $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
6: return μ_t, Σ_t

This algorithm updates the state by performing two steps, namely the prediction step and the update step. The former includes Instructions 1-2 and predicts the next state $\hat{\mu}_t$ given the current state μ_t according the evolution described a motion model $g(u_t, \mu_{t-1})$ affected by a Gaussian noise $\epsilon_t \sim \mathcal{N}(0, R_t)$. Since the EKF algorithm works with linear models, the motion model is approximated to the first order derivative evaluated in the robot state. The Gaussian noise Σ_t is updated with the Jacobian G_t in the Instruction 2, which also accounts for the presence of a Gaussian noise R_t, also called process noise. In the EKF-SLAM algorithm these steps are the same, with the exception that the state of the filter contains both the robot pose estimate and the landmarks position estimate. We have chosen as motion model the odometry motion model [4] which describes each movement as three consecutive phases according to Figure 1. The first type of movement is a rotation δ_{rot1} which rotates the robot from its original position θ towards the direction of movement. The second, δ_{trans} realizes the shift in position, the third, δ_{rot2}, instead, rotates the platform again to reach the final orientation. Mathematically, the robot state is updated as illustrated in Equation 2.

$$\begin{aligned} x_{t+1} &= x_t + \hat{\delta}_{trans} cos(\theta^t + \hat{\delta}_{rot1}) \\ y_{t+1} &= y_t + \hat{\delta}_{trans} sin(\theta^t + \hat{\delta}_{rot1}) \\ \theta_{t+1} &= \theta_t + \hat{\delta}_{rot1} + \hat{\delta}_{rot2} \end{aligned} \tag{2}$$

The rotations and translations here are intended as affected by a Gaussian noise dependent on the space traversed by the robot. For the application to EKF-SLAM algorithm we separated the

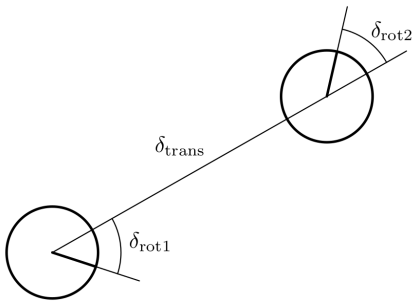

Figure 1: The tree type of movement composing the odometry motion model: a rotation δ_{rot1}, a transition δ_{trans} and a final rotation δ_{rot2} (Figure 5.7 in [4])

update of the state from the noise:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \delta_{trans} cos(\theta_t + \delta_{rot1}) \\ \delta_{trans} sin(\theta_t + \delta_{rot1}) \\ \delta_{rot1} + \delta_{rot2} \end{bmatrix} \tag{3}$$

where δ_{rot1}, δ_{trans} and δ_{rot2} are computed over the control input, which is represented by the odometry measurement. The Gaussian noise is later added through the process noise matrix R_t, a diagonal matrix which contains the variances fixed over x, y and θ:

$$R_t = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{4}$$

During the correction step of the algorithm, Instruction 3-5 in Algorithm 1, the landmarks are used to perform a correction over the prediction. However, the plants detected from the plant detection phase can not be used directly as their position is expressed in sensor coordinates, but a conversion is needed to store them in the map coordinates. The inverse sensor model takes care of this task:

$$\begin{bmatrix} x_m \\ y_m \end{bmatrix} = \begin{bmatrix} cos(\theta_t) & -sin(\theta_t) \\ sin(\theta_t) & cos(\theta_t) \end{bmatrix} \begin{bmatrix} x_s \\ y_s \end{bmatrix} + \begin{bmatrix} x_R \\ y_R \end{bmatrix} \tag{5}$$

Whenever a new plant is detected, its position in the sensor coordinate frame is converted to the map coordinate frame to be checked against all the registered landmarks and see whether it is already present or not. The landmark returned, if any, would then be used during the update phase in the computation of the matrix H_t. If no landmark is selected, then the new landmark is

added to the filter state. Since many landmarks are sighted by the sensor at a time, instead of updating the filter many times, we can update it once by computing different matrices H_i by filling only those fields relative to the robot pose and the landmark i, leaving as zeroes all the others, and by stacking them vertically to form an unique matrix H_t. To compute the error involved in the update, the returned matching landmark is converted into the sensor coordinate frame through the sensor model:

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} cos(\theta_t) & sin(\theta_t) \\ -sin(\theta_t) & cos(\theta_t) \end{bmatrix} \left(\begin{bmatrix} x_m \\ y_m \end{bmatrix} - \begin{bmatrix} x_t \\ y_t \end{bmatrix} \right) \tag{6}$$

To take into consideration the uncertainties during the matching, we performed a χ-square test between the entry point and all the stored landmarks. This test computes the distance of a point to a multivariate Gaussian using the Mahalanobis distance. Since the Mahalanobis distances of a Gaussian of N degrees of freedom, in this case 2, follows the χ-square distribution of the same degrees of freedom, we run this test by comparing the Mahalanobis distance to a critical value of the distribution, selected based on a confidence level. A distance over the threshold would result in the two measurements being marked as different, while below they would be associated. The first landmark passing the test is returned.

3.3. Plant counting

The map produced in the previous step is post-processed to assess the number of plants. We've decided to wait for the map to be complete before starting the effective counting as there is the possibility of the EKF-SLAM algorithm to detect the same plant twice because of failure during the landmark association process. Therefore, we used the distance between one plant and the other to mark a circular perimeter within which any landmark present is considered as detection for the same plant. The nearest landmark to the plant position is then kept while the others are discarded.

4. Experimental results

We conducted the evaluation of our algorithm both in simulation and in a real world scenario by using an external dataset recorded in a greek vineyard [3]. Our purpose was to find the parameters for the EKF-SLAM algorithm in the simulation and porting them to the real case.
As for the former, we built a world in a 3D simulation software, Gazebo, to try to resemble the environment of a real orchard. We mounted the 3D LiDAR over a robotic platform and we drove it along the plant rows. Since a noisy source of odometry is not available in Gazebo, we created one by exploiting the full odometry motion model, which, with respect to the one previously described, adds Gaussian noises to each of the computed movement:

$$\begin{aligned} \hat{\delta}_{rot1} &= \delta_{rot1} - \mathcal{N}(0, \alpha_1\delta^2_{rot1} + \alpha_2\delta^2_{trans}) \\ \hat{\delta}_{trans} &= \delta_{trans} - \mathcal{N}(0, \alpha_3\delta^2_{trans} + \alpha_4\delta^2_{rot2}) \\ \hat{\delta}_{rot2} &= \delta_{rot2} - \mathcal{N}(0, \alpha_1\delta^2_{rot2} + \alpha_2\delta^2_{trans}) \end{aligned} \tag{7}$$

For assessing the performance of the system, we run different tests using each time a different noisy trajectory. We counted the number of true positives, i.e. the plant correctly detected, false positives, i.e. the detection which did not correspond to any plant and the false negatives, i.e. the plants not marked by any landmark. The threshold for true positives was marked by half the distance between one plant and the successive one in the row. Then, we computed precision and recall metrics, assessing the number of plants rightfully registered by the algorithm over the total amount of detections and the total number of plants in the field respectively. These metrics have been reassessed also in post-processing, by discarding the landmarks detected as double-counting for the same plant. Also, we added to the aforementioned metrics the mean average error (MAE) in meters and the Root Mean Squared Error (RMSE) in meters, to account for the average distance between the detections and the ground truth and the spreadness of the measurements. Finally, we calculated the mean distance of the estimated trajectory from the real one traced by the followed. Although the plane model was detected well, the average results, reported in table [**?**] show that the presented system is not accurate as the ones reviewed in literature, which, however did not perform SLAM. By observing the details of each run, the algorithm demonstrated overall to be too sensitive in the changes in the type of noise

Precision	0.65
Recall	0.70
Precision (Post-process)	0.69
Recall (Post-process)	0.7
MAE (m)	0.174
RMSE (m)	0.15
Mean distance (m)	0.298

Table 1: Average results obtained from the simulation

applied during the estimation, ranging from almost perfect counting with 77 plants detected over 78 to 45 or less plant detected. However, it demonstrated a good performance by scoring more than 90% in precision and recall in the run of a noise similar to the one registered by the visual odometry in the Greek environment.

Instead, regarding the testing over the real dataset, he proposed algorithms could be evaluated only by performing a visual comparison between trajectories because of the unavailability of a field map and impossibility to estimate one. We used as source of odometry the visual odometry computed by a ZedNode present on the robot and as ground truth a GPS. The robot traversed the field in different rows, but we used just the first one (forward-turn-turn-backward) to assess the system performances. We slightly adjusted the parameters found in the simulation to account for a major noise in visual odometry. The algorithm detected poles and plants as belonging to the same category since the plant detection phase can not distinguish between different classes. Also, the detection of some of them failed because of the resolution provided by the employed LiDAR which captured the plants with a number of points inferior to the one used in simulation. The EKF-SLAM algorithm managed to correct the length in the first straight line traversed which was underestimated by the ZedNode, but not the curve in the odometry trajectory. However, it mismatched landmarks when performing turns, thus derailing from the ground truth trajectory when turning back to the start point.

5. Conclusions

We proposed an approach to plant counting by exploiting an EKF-SLAM approach, which, however, scored less than the solutions present in literature. Future developments of such algorithm might include improvements in the plant detection task, with the distinction between plant and non-plant elements in the environment and possibly with the assessment of plant health status. Also, the SLAM algorithm might find benefits from the recognition of rows of plants, instead of single plants for the correction of the trajectory, which can overcome the limitations showed by the testing in the real world scenario.

References

[1] Erkan Kayacan, Zhongzhong Zhang, and Girish Chowdhary. Embedded high precision control and corn stand counting algorithms for an ultra-compact 3d printed field robot. 06 2018.

[2] Joe Luck, Santosh Pitla, and Scott Shearer. Sensor ranging technique for determining corn plant population. 06 2008.

[3] Riccardo Polvara, Sergi Molina, Ibrahim Hroob, Alexios Papadimitriou, Tsiolis Konstantinos, Dimitrios Giakoumis, Spiridon Likothanassis, Dimitrios Tzovaras, Grzegorz Cielniak, and Marc Hanheide. Blt dataset: acquisition of the agricultural bacchus long-term dataset with automated robot deployment. *Journal of Field Robotics, Agricultural Robots for Ag 4.0*. Under Review.

[4] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[5] Ulrich Weiss and Peter Biber. Plant detection and mapping for agricultural robots using a 3D LIDAR sensor. *Robotics and Autonomous Systems*, 59(5):265–273, May 2011.