Robot Navigation in Orchards Using Top-View Imagery

Omer Shalev
Robot Navigation in Orchards Using Top-View Imagery

Research Thesis
In Partial Fulfillment of the Requirements for the Degree of Master of Science in Autonomous Systems and Robotics

Omer Shalev

Submitted to the Senate of the Technion - Israel Institute of Technology
Sivan 5779 Haifa June 2019
The Research Thesis Was Conducted Under the Supervision of Assoc. Prof. Amir Degani, in The Interdepartmental Program of Autonomous Systems and Robotics

List of Publications

Parts of this thesis have been presented in conferences during the course of the author’s research period:


Parts of this thesis are being processed to an article:

Table of Contents

1 Abstract .......................................................................................................................... 1
2 Introduction and Related Work .................................................................................. 3
  2.1 Common Approaches .............................................................................................. 3
    2.1.1 Baseline Performance ...................................................................................... 5
  2.2 Thesis Outline ......................................................................................................... 8
3 Experimental Setup ..................................................................................................... 10
4 Computer Vision Methodology .................................................................................... 12
  4.1 Canopies Extraction ............................................................................................... 14
  4.2 Trunks Approximation and Semantic Labeling ..................................................... 20
    4.2.1 Image Alignment ............................................................................................. 44
5 Navigation Architectures ............................................................................................... 47
  5.1 Low Altitude ........................................................................................................... 48
    5.1.1 Canopy-Based AMCL ..................................................................................... 50
    5.1.2 Canopy-Based ICP ......................................................................................... 67
  5.2 High Altitude .......................................................................................................... 69
    5.2.1 UGV Periodic State Update ........................................................................... 69
    5.2.2 Semantic Global Path Planning ....................................................................... 72
6 Summary and Discussion ............................................................................................... 78
  6.1 Future Work ............................................................................................................ 78
7 Bibliography ................................................................................................................ 80
8 Appendices ................................................................................................................... 83
  9.1 The Negligible Effect of the Keystone Distortion .................................................... 83
  9.2 Trunks Approximation and Semantic Labeling – Additional Results .................. 84
  9.3 Pixels to Meters Scaling ......................................................................................... 88
  9.4 Parameter Values .................................................................................................... 89
List of Figures

Figure 1: Rough sketch of the UGV's trajectory .......................................................... 5
Figure 2: Raw wheel odometry estimation ................................................................. 6
Figure 3: EKF estimation ......................................................................................... 6
Figure 4: ICP estimation ......................................................................................... 7
Figure 5: gmapping output ....................................................................................... 7
Figure 6: thesis outline ........................................................................................... 9
Figure 7: Clearpath Jackal with SICK LMS111 ....................................................... 10
Figure 8: DJI Mavic Air (taken from photographylife.com) ..................................... 11
Figure 9: UGV from 80 meters ............................................................................... 12
Figure 10: UGV from 20 meters ............................................................................ 13
Figure 11: compact orchard plot representation .................................................... 13
Figure 12: mask of green areas .............................................................................. 15
Figure 13: contours extracted using Suzuki and Abe’s Border Following algorithm ....................................................................................... 15
Figure 14: canopies mask and canopy contours following area filtering ............... 16
Figure 15: canopy contours on RGB image .............................................................. 16
Figure 16: canopy contours comparison ................................................................. 17
Figure 17: contours of four aligned images (April) .................................................. 18
Figure 18: contours of five aligned images (April in green and November in purple) ....................................................................................... 19
Figure 19: trunks approximation and semantic labeling (April, 15:08) .................... 20
Figure 20: trees layout of the plot in Figure 19 ........................................................ 21
Figure 21: 6×6 tree grid .......................................................................................... 21
Figure 22: transformed grid parameters ................................................................... 22
Figure 23: the central cropped region of the image ................................................ 24
Figure 24: canopies mask of the cropped image ...................................................... 24
Figure 25: rotated masks with their column sums vectors ........................................ 26
Figure 26: column sums minima determine inter-row passage centers ................. 28
Figure 27: row sums maxima determine centroids’ vertical positions in image slice ....................................................................................... 29
Figure 28: estimated tree centroids marked on the image ...................................... 30
Figure 29: “drift vectors” between centroids ............................................................ 31
Figure 30: “drift vectors” following outlier removal ................................................ 32
Figure 31: optimal placement of the 6×6 grid (blue points) on the estimated centroids (red points) ............................................................... 33
Figure 32: grid of cropped gaussians ....................................................................... 34
Figure 33: pixelwise multiplication of the canopies reward mask with the grid of gaussians (positive values only) ................................................................. 34
Figure 34: selected stages in the Nelder-Mead optimization process ....................... 37
Figure 35: the optimized 6×6 grid (green) vs the initial estimated 6×6 grid (blue) ....................................................................................... 38
Figure 36: the optimized grid extrapolated on the entire image ............................... 39
Figure 37: normalized tree scores of the extrapolated grid vertices ....................... 40
Figure 38: optimal vs sub-optimal placement of the layout ..................................... 41
Figure 39: trees layout matched to the image with semantic labeling ....................... 42
Figure 40: image alignment using approximated trunk positions ......................... 45
Figure 41: bad alignment of 15:08 to 19:04 (April) using the typical ORB method ....................................................................................... 46
Figure 42: virtual canopies scan (red) around an origin point (purple) .................... 49
List of Tables and Algorithms

Algorithm 1: Extract-Canopy-Contours ................................................................. 14
Algorithm 2: Extract-Canopies-Mask ................................................................. 14
Algorithm 3: Approximate-And-Label-Trunks .................................................... 23
Algorithm 4: Estimate-Tree-Rows-Orientation ..................................................... 25
Algorithm 5: Estimate-Tree-Centroids ................................................................. 27
Algorithm 6: Estimate-Trees-Grid-Cell-Dimensions ........................................... 30
Algorithm 7: Estimate-Trees-Grid-Shear ............................................................. 31
Algorithm 8: Find-Grid-Placement ....................................................................... 33
Algorithm 9: Calculate-Tree-Score ...................................................................... 35
Algorithm 10: Calculate-Grid-Score ...................................................................... 35
Algorithm 11: Optimize-Trunks-Grid ................................................................. 36
Algorithm 12: Align-Layout-On-Extrapolated-Grid .......................................... 39
Algorithm 13: Refine-Trunk-Positions .................................................................. 43
Table 14: similarity of selected pairs of images .................................................. 46
Algorithm 15: Generate-Virtual-Canopies-Scan ............................................... 49
Table 16: aggregated AMCL results (April, plot #1) ............................................ 59
Table 17: aggregated AMCL results (November, plot #1) ..................................... 59
Table 18: aggregated AMCL results (November, plot #2) ..................................... 61
Table 19: aggregated AMCL results (measurement noise) ..................................... 63
Table 20: aggregated AMCL results (slippage noise) ......................................... 65
Table 21: aggregated AMCL results (drift noise and constant slippage noise) ........ 66
Table 22: final EKF estimation errors ................................................................. 70
Algorithm 23: Get-Adjacent-Nodes-Distances ...................................................... 73
1 ABSTRACT

Mobile robots are becoming common in agriculture and are used for a variety of purposes. In orchards, there are various tasks that a mobile robot can perform, such as sensing plant stress, sensing pests, yield monitoring, or selective spraying. One commonality to these tasks is the need for autonomous and accurate navigation. While for some tasks it is enough to have a rough location estimate, for others it is crucial to have centimeter-level accuracy to allow precise sensing or precise manipulation capabilities, i.e., precision agriculture. As examples, accurate localization of an Unmanned Ground Vehicle (UGV) allows to execute accurate and selective pesticide spraying based on the tree’s status and history. The ultimate objective of this thesis is to provide navigation solutions which can serve for precision agriculture use cases.

Navigation of ground vehicles in orchards is a complex problem which is yet to be fully addressed. The typical navigation approaches are not adjusted to the characteristics of the orchard environment, such as large dimensions, difficult terrain and homogeneous scenery. In addition, Global Positioning System (GPS) localization is usually not applicable in orchards due to signal occlusions. To alleviate the above-mentioned difficulties in this complex environment, we propose to use top-view images of the orchard acquired in real time. This additional auxiliary sensing aids by providing additional information to the ground vehicle.

Our navigation approaches rely on computer vision techniques that are applied on the top-view images. By extracting “canopies masks” from the images, we form a heterogeneous and compact representation of the orchard. These techniques also allow to form a semantic tree map which distinguishes between the individual trees and labels them. Using these representations, we are able to tackle the navigation challenges in new ways which are GPS-independent and refrain from the use of artificial landmarks.

In this work, we suggest two families of applicable navigation architectures that leverage the top-view observations from different altitudes. For continuous low altitude video streams, we present an innovative way to deal with the “kidnapped robot problem” by using the canopies masks extracted from the images. Using high altitude images, we propose a semantic global path planner which plans trajectories between the labeled trees and is based on a cost map derived from the canopies mask. As these high altitude images are acquired periodically and opportunistically, we suggest using them also for periodic pose updates of the ground vehicle.

The proposed approaches are supported by field experiments conducted in several real orchards, during different seasons and different times of the day. The data collected at the field was used in numerous offline experiments and analyses. These experiments demonstrate the effectiveness of our suggested approaches, both in terms of accuracy and repeatability.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMCL</td>
<td>Adaptive Monte Carlo Localization</td>
</tr>
<tr>
<td>D-GPS</td>
<td>Differential Global Positioning System</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HSV</td>
<td>Hue Saturation Value</td>
</tr>
<tr>
<td>ICP</td>
<td>Iterative Closest Point</td>
</tr>
<tr>
<td>IID</td>
<td>Independent and Identically Distributed</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurements Unit</td>
</tr>
<tr>
<td>LiDAR</td>
<td>Light Detection and Ranging</td>
</tr>
<tr>
<td>MCL</td>
<td>Monte Carlo Localization</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Square Error</td>
</tr>
<tr>
<td>RGB</td>
<td>Red Green Blue</td>
</tr>
<tr>
<td>RTK</td>
<td>Real-Time Kinematic</td>
</tr>
<tr>
<td>SLAM</td>
<td>Simultaneous Localization and Mapping</td>
</tr>
<tr>
<td>SSIM</td>
<td>Structural Similarity Index Measure</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>UGV</td>
<td>Unmanned Ground Vehicle</td>
</tr>
</tbody>
</table>
With an expected increase rate of 30% in the world’s population by 2050 [1], accelerated urbanization trends [2], and shortage of labor power for manual work – even in third world countries [3] – there is a growing gap between the rapidly rising demand for food and other agricultural products and the ability of farmers to expand production. To address the upcoming demand challenges, agricultural efficiency must be increased. For the most part, automation increases the efficiency of repetitive agricultural procedures such as harvesting in field crops [4]. However, automation is limited to well-defined use cases and thus requires a large degree of human involvement. Precision agriculture is considered the next evolution of agricultural optimization and involves perception, understanding and decision making [5]. While automation operates on a group of non-distinct individual elements (e.g., plants or fruits), precision agriculture provides differential treatment to every individual. Robots play a key role in many precision agriculture applications being the means of sensing, processing and acting.

Our focus in this study is precision agriculture in orchards, where the individual elements are trees. In orchards, precision agriculture tasks that can be executed by mobile robots are numerous, ranging from disease and pest identification, spraying, irrigation leaks identifications, yield monitoring, to harvesting and post harvesting tasks. One communality which the above tasks share is the need to continuously obtain an accurate position of the robot in the orchard. The localization problem in orchards, as well as its sibling problems, namely mapping and path planning, are the focus of this work.

2.1 Common Approaches
The localization and mapping of mobile robots were extensively studied in recent years. Mobile robots are typically equipped with various perception means such as wheel encoders, inertial sensors, GPS, lasers scanners, sonar scanners, cameras, etc., and those are used by localization algorithms to estimate the robot’s location. Simultaneous Localization and Mapping (SLAM) algorithms not only estimate the robot’s location but also build and update a map of the robot’s environment [6]. The map is used as means of feedback for the location estimator and vice versa.

Different navigation environments of mobile robots have different characteristics. The environment dimensions, the terrain and the visual scenery can be utterly different, leading to different behavior of the robot’s sensors. Thus, nowadays’ localization and mapping approaches are not generic enough and there are still branches of these problems which have yet been solved.

Orchard navigation environments present unique challenges for localization and mapping. First, the terrain roughness leads to significant vehicle skid. Additionally, the large dimensions of the orchard lead to large accumulated error in the estimated pose and therefore, wheel odometry [7] is not a reliable solution for UGV localization (as demonstrated in chapter 2.1.1). Another common approach for localization in outdoor environments, which is extensively used in open field crops [8], is the use of GPS. In orchards, GPS has several drawbacks which impair its applicability. First, GPS depends on satellite coverage, which deteriorates at the ground level due to occlusions from the tree canopies [9], [10]. Second, the use of GPS alone does not allow to estimate the robot’s orientation and requires an additional sensor such as an Inertial Measurements Unit (IMU) or a compass. Third, the nominal GPS accuracy of approximately five meters is insufficient for precision agriculture tasks. Differential GPS (D-GPS) or Real-
Time Kinematic (RTK) GPS offer nominal accuracy of approximately ten centimeters but require a constant base station in the field and are rather expensive. In addition, D-GPS and RTK-GPS suffer from similar occlusion problems in orchards as mentioned above, and hence their accuracy deteriorates in such environments. Finally, orchards are also problematic for visual odometry [11] since leaves and trunks lack unique visual signatures, and hence no natural landmarks allow loop closure of graph-based visual SLAM algorithms [12].

A backbone of many previous works related to GPS-denied localization in orchards is the Extended Kalman Filter (EKF) that is used to fuse information from different information channels such as wheel odometry, IMU readings, Light Detection and Ranging (LiDAR) scans and camera streams. A typical approach leverages the orchard’s geometric characteristics and uses a compact map representation of tree lines. Two-dimensional LiDAR, mounted on a UGV, provides wide horizontal scans of the tree trunks level and of artificial landmarks (e.g. reflective tapes) that are used in some of the works, e.g., [13]. Computer vision techniques such as Hough transform [14] can perceive the tree lines which are used as measurements for the EKF that localizes the robot on the map built a priori [13], [15], [16]. This approach is satisfactory for simple tasks that only involve patrolling and require rough location estimate. One major drawback of this approach is the fact it uses the robot’s starting point as a global reference. For the kidnapped robot problem, i.e., estimating the robot’s pose when the initial pose is unknown (e.g., following hardware reset), this approach is unsuitable.

Other works that focus on orchard mapping vastly rely on a three-dimensional point cloud acquired by a LiDAR or stereo camera. Several among these works tackle the mapping problem by detecting artificial landmarks that are used for loop closure [17] while others detect and segment trees using various techniques [18], [19], [20]. All these mapping methods require expensive perception sensors and strong computational power. GPS, or even D-GPS, is also required in these works for localization during the mapping stage. Moreover, the tree segmentation algorithms use certain heuristics about tree shape and hence are typically environment dependent.

Monte Carlo Localization (MCL) [21] and its commonly used variant, Adaptive Monte Carlo Localization (AMCL) [22], are probabilistic localization approaches that use a particle filter to track the pose of a robot against a known map based on sensor readings. The big advantage of these approaches is their ability to operate without prior knowledge of the robot’s initial state, that is to say, they address the kidnapped robot problem. Being based on a particle filter, MCL and AMCL are likely to converge in environments where the heterogeneity is large. As perceived by most sensing means mounted on a UGV, the heterogeneity of an orchard is low due to the similar trunk sizes and spacing and thus, orchards are traditionally conceived unsuitable for MCL and AMCL.

Recent works related to path planning in orchards mainly focus on patrolling. A typical approach guides an autonomous vehicle to traverse the orchard rows sequentially while maintaining the row centers by means of tree sensing [15]. Semantic path planning, in which a vehicle is guided to specific trees of interest, requires a semantic tree map in which individual trees are labeled and positioned. Such semantic maps are not achievable using standard perception techniques at the ground level (Figure 5).

Along with the significant advances of computer vision algorithms in the recent years, the use of imaging in agriculture has also been extensively studied. Images acquired at the ground level by cameras mounted
on UGVs are used for various detection and classification purposes at a local scale, e.g. [23], [24], [25]. Top-view images are also used for similar purposes but at a global scale, e.g. [26].

2.1.1 Baseline Performance
To better substantiate the difficulties with the common navigation approaches in orchards, we tested several typical algorithms in the field by slowly teleoperating a UGV along the trajectory roughly sketched in Figure 1. The trajectory starts and ends at the same point (marked in red).

The raw wheel odometry of the UGV is illustrated in Figure 2. Its errors mainly stem from incorrect heading estimation, as is evident in all turning points.

![Figure 1: Rough sketch of the UGV's trajectory](image)
We examined an EKF which fuses the raw odometry together with IMU readings. The estimated trajectory is illustrated in Figure 3. Despite significant improvement relative to the raw odometry (Figure 2), the accumulated error at the end point is still significant (about 15 meters) and thus we conclude that this approach is unsuitable for centimeter-level accuracy which is required for precision agriculture applications.

We also examined an alternative common odometry source which is LiDAR based. The Iterative Closest Point (ICP) [27] algorithm operates on readings of the LiDAR mounted on the UGV. The ICP output is illustrated in Figure 4. ICP does not bring significant improvement compared with the wheel odometry (Figure 2).
In addition to state estimation, we also examined mapping. We used the same LiDAR readings mentioned above as inputs to the popular gmapping algorithm [28]. Figure 5 illustrates the generated map which is obviously unsatisfying.

Figure 4: ICP estimation

Figure 5: gmapping output
2.2 Thesis Outline

In this study we propose to deal with localization, mapping and path planning of a single UGV in an orchard with the aid of top-view images. We consider the top-view as complementary to sensors mounted on the UGV which are responsible for the local perception and planning.

In chapter 4 we present the computer vision techniques which are used in order to generate a compact representation of the orchard in the form of a canopies mask and approximated trunk positions (Figure 6-A). Apart from being compact and efficient from computational perspective, we show that this representation is consistent and robust to varying wind and light conditions.

In chapter 5 we propose several applicable navigation architectures for leveraging top-view images in robotic systems in a way that regards engineering considerations. We differentiate between low altitude video streams in which the UGV is continuously tracked, and high altitude snapshot images acquired opportunistically and at lower frequency.

The top-view observation of the orchard provides a unique signature of every tree formed by the shape of its canopy. This practically changes the homogeneity premise in orchards and also paves the way for addressing the kidnapped robot problem. In chapter 5.1 we present the concept of virtual canopies scan which is generated from low altitude video streams. We apply AMCL on this virtual scan to localize the ground vehicle against one high altitude image which is used as a map (Figure 6-B). One noteworthy achievement in this thesis is the demonstration of centimeter-level localization accuracy with this approach. This improved accuracy is essential for precision agriculture use cases. In chapter 5.1.2 we present another potential use of the virtual canopies scans for incremental positioning using ICP.

The approximation of trunk positions allows us to align different high altitude images (Figure 6-C) of the same orchard plot. Using this alignment method (elaborated in chapter 4.2.1), we demonstrate in chapter 5.2.1 how opportunistic updates of the ground vehicle’s EKF from the top-view observation improve its pose estimation. Additionally, in chapter 5.2.2 we use the top-view images to build cost maps which we use to plan semantic trajectories (Figure 6-D) and to avoid obstacles.

Being an applicative study, this work demonstrates the suggested navigation architectures on data acquired in field experiments. Part of these architectures are demonstrated as a proof-of-concept; other architectures are extensively examined on various data samples and conditions. The approaches we examine are all GPS independent and do not require the use of artificial landmarks in the orchard.
A: compact representation is generated using computer vision techniques

B: low altitude video is used for localization against a high altitude map

C: matched trunk positions allow to align different high altitude images

D: cost map allows to plan paths between semantic source and target

Figure 6: thesis outline
3 Experimental Setup

The concepts presented in this study are all examined on real data acquired in field experiments. We chose to focus on almond (Prunus dulcis) orchards, and the final experiments were conducted in Kibbutz Lavi in northern Israel. In order to strengthen the reliability of our findings, we collected multiple data samples at different hours of the day and at different seasons: April and November. In April, the data was collected in one orchard plot throughout the day; in November, we collected data at the same original plot and in an additional plot.

Our system hardware included one UGV and one Unmanned Aerial Vehicles (UAV). For the ground robot, we used the Clearpath Jackal UGV [29]. Relying on a high torque 4x4 drivetrain, the Jackal overcomes the rough outdoor terrains and is thus suitable for driving in orchards. In addition, Jackal is ROS [30] compatible and allows easy integration with existing navigation software libraries. In addition to the onboard sensors, we integrated on top of the Jackal a Microstrain 3DM-GX4-25 [31] IMU and a SICK LMS111 [32] LiDAR scanner. To alleviate the visual detection of the Jackal in the top-view images, we attached a colored marker on the upper surface of the UGV (Figure 7).

The DJI Mavic Air [33] (Figure 8) was used as the aerial robot. With up to 21 minutes flight time and a 4K camera operating at 30 Hz and mounted on a 3-axis gimbal, the Mavic Air matched our experimental requirements. Mavic Air has proved to be stable in outdoor conditions, even in strong winds. Both robots were teleoperated by a human operator and the collected data was saved as ROS bag files (UGV telemetry and sensor readings) and JPG or MP4 files (UAV camera).

Figure 7: Clearpath Jackal with SICK LMS111
Four bright blue markers were placed in the orchard plot, but those served only as reference for the plot’s location and boundaries and were never used in any of the algorithms as landmarks. In some of the experiments we placed a white foldable table that was used to mimic an obstacle for the ground robot.

The collected data was used in multiple offline experiments. The offline work allowed to test the suggested algorithms multiple times and in a wide range of conditions. Some of the experiments incorporated synthesized noises on top of the original data in order to stress-test the algorithms.

The offline experiments were executed on a desktop PC with i7-6700 CPU and 16 GB RAM running Ubuntu 16.04 and ROS Kinetic. Computer vision code was implemented in Python using OpenCV [34] and NumPy [35]. Localization and mapping algorithms mainly relied on ROS libraries. Automation scripts were developed in Python and allowed dozens of hours of continuous experiments execution.

Entire code used in this work can be found in our GitHub repository https://github.com/CearLab/orchards_top_view_aided_navigation_experiments.
4 COMPUTER VISION METHODOLOGY

The key concept of this study is the use of top-view images for achieving improved navigation of a ground vehicle in an orchard. For this purpose, the top-view image must be acquired at an altitude which allows detecting the ground vehicle in the image. Figure 9 and Figure 10 are examples of such images, acquired at 80 and 20 meters respectively. The 80 meters image is advantageous in capturing the entire orchard plot while the 20 meters image is advantageous in getting a more detailed perception of the vehicle’s close vicinity.

In this work we developed two computer vision procedures that are applied on the top-view images: canopies extraction and trunks approximation. These two procedures provide a compact representation of the orchard and practically reduce the dimensions of representation from a two-dimensional RGB image to a binary mask of canopies and a set of points representing the approximated trunk positions (exemplified in Figure 11). This compact representation is naturally advantageous in terms of computational efficiency. This representation also paves the way for innovative ways of dealing with localization, mapping and path planning tasks in orchards, as later described in chapter 5.

Figure 9: UGV from 80 meters
Figure 10: UGV from 20 meters

Figure 11: compact orchard plot representation
4.1 Canopies Extraction

The basic computer vision procedure used in this study is the segmentation of canopies from the top-view images. A canopy contour is defined as the boundary of an area projected from the three-dimensional tree to the ground level. In Algorithm 1 we convert the image to Hue Saturation Value (HSV) representation. This allows us to extract the green canopy areas from the image and create a binary green mask (Figure 12). From this mask, the contours are extracted using the Border Following algorithm suggested by Suzuki and Abe [36] (Figure 13). To remove outliers, we calculate the interior area of every contour and those below a threshold are removed (Figure 14). Figure 15 visualizes the final result of contours extraction on the original RGB image used for the previous three figures. It is evident that there are still outliers among the extracted contours, especially in non-canopy green areas of the image (e.g., weeds outside the orchard). Our navigation applications which use this procedure will have to assume imperfect canopies extraction. In most of our navigation applications, we rather use the canopies mask representation, formed by the contour interiors, as described in Algorithm 2.

Algorithm 1: Extract-Canopy-Contours

<table>
<thead>
<tr>
<th>Input:</th>
<th>image [RGB matrix]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>contours_list [array of point sets]</td>
</tr>
<tr>
<td>Constants:</td>
<td></td>
</tr>
<tr>
<td>• UPPER_HSV_BOUND</td>
<td></td>
</tr>
<tr>
<td>• LOWER_HSV_BOUND</td>
<td></td>
</tr>
<tr>
<td>• CONTOUR_AREA_THRESHOLD</td>
<td></td>
</tr>
</tbody>
</table>

image := Convert-To-HSV(image)
green_mask := Pixelwise(LOWER_HSV_BOUND ≤ image ≤ UPPER_HSV_BOUND)
contours_list := Border-Following(green_mask)
for contour in contours_list:
    if Area(contour) < CONTOUR_AREA_THRESHOLD:
        remove contour

Algorithm 2: Extract-Canopies-Mask

<table>
<thead>
<tr>
<th>Input:</th>
<th>image [RGB matrix]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>canopies_mask [binary matrix]</td>
</tr>
</tbody>
</table>

contours_list := Extract-Canopy-Contours(image)
canopies_mask := Empty-Matrix()
for contour in contours_list:
    canopies_mask := Draw-Contour-Interior-On-Image(canopies_mask, contour)
Figure 12: mask of green areas

Figure 13: contours extracted using Suzuki and Abe’s Border Following algorithm
Figure 14: canopies mask and canopy contours following area filtering

Figure 15: canopy contours on RGB image
The canopy contours are intended to be the trees’ signatures. In order to serve for localization and mapping tasks, the variations in those signatures between images must be minimal. One factor which might alter the canopies shape throughout the day is winds. However, the top-view observation smooths the variations in leaf movements caused by the wind. Another factor which might affect the consistency of contour signatures is the shade direction, but our segmentation algorithm distinguishes between shade and canopies in most cases. Figure 16 illustrates the canopy contours on the same zoomed-in area of the orchard in four images acquired at different times of the day in April and in one image acquired in November.

Figure 16: canopy contours comparison
Even though there are clear differences between the four April images, which mainly stem from the different lighting conditions, the overall shape of the canopy contours is similar in all four. Even the outlier weeds are consistently extracted as a canopy in all of them. However, the shape of contours on the November image are significantly different than the ones on April images.

Figure 17 overlays the aligned contours of all four April images used in Figure 16. The alignment is performed using the method described in chapter 4.2.1. In Figure 18, those four images are also aligned with the November contours. These illustrations provide an indication of the consistency of our canopy contours extraction method for the short term (hours or possibly several days). For the long term however, the variations in canopy contours’ shapes are significant. This implies that a new reference image must be captured on a daily or weekly basis.

Figure 17: contours of four aligned images (April)
Figure 18: contours of five aligned images (April in green and November in purple)
4.2 Trunks Approximation and Semantic Labeling

As stated above, a mandatory capability for precision agriculture in orchards is the distinction between individual trees. In this section we present the method we developed for approximating trunk positions and semantically labeling them from high altitude top-view images (exemplified in Figure 19).

Our approach relies on a pre-defined trees layout of the orchard plot that the farmer provides. While in some cases there is a physical separation between plots (e.g., wide gap from adjacent plots), in other cases this distinction is not well-defined, and it stems from considerations such as trees’ age. The layout structure provides flexibility to the farmer to define the preferred partitioning of the orchard plots. The layout is virtually a tree grid, represented as a matrix whose dimensions determine the plot’s boundaries and whose values describe the grid vertices. The three matrix’s allowed values are:

- ‘+1’: a tree is present in this grid vertex
- ‘-1’: a tree is necessarily not present in this grid vertex
- ‘0’: a tree might be present in this grid vertex, but it does not belong to this plot

Figure 20 describes the layout corresponding to the labeled trunks in Figure 19.

![Figure 19: trunks approximation and semantic labeling (April, 15:08)](image-url)
Figure 20: trees layout of the plot in Figure 19

The trunks approximation procedure assumes three basic assumptions concerning the orchard plot and its top-view image:

1. All trees in the orchard plot matching the provided layout appear in the image. Trees from adjacent plots might appear as well, but it is assumed that the majority of the trees in a center crop of the image belong to that plot.
2. The trees are planted in parallel rows such that inter-row distances are approximately even. The intra-row distances between trees should also be approximately even. In other words, the general structure of the trees’ placement, as observed in the top-view image, is an affine transformation of a rectangular grid (Figure 21). We assume an affine transformation and not a complete homographic transformation since the Keystone distortion is negligible (as demonstrated in Appendix 9.1).
3. The plot is big enough to include at least one complete 6×6 tree grid.

Figure 21: 6×6 tree grid
The trunks approximation procedure is comprised of three essential stages:

1. Fitting a $6 \times 6$ affine-transformed tree grid to a specific area in the image. We estimate four parameters of the grid: the grid cell dimensions ($\Delta x, \Delta y$), the grid orientation ($\theta$) and the grid shear ($\alpha$), as illustrated in Figure 22. In addition, we find the placement ($O_x, O_y$) of the $6 \times 6$ grid on the image such that it best fits a certain group of $6 \times 6$ trees in the image. The advantage of fitting a grid of trees rather than estimating tree positions individually is the averaging effect which compensates for badly extracted trees.

2. Extrapolating the fitted $6 \times 6$ grid on the entire image and aligning the plot layout on it. In other words, in this stage we do not approximate trunk positions but rather detect the placement of the plot on the image. At this stage we also label the trees with semantic names according to the provided layout.

3. Refining the matched trunk positions to fit the image better. Bearing in mind that the plot is not a perfectly transformed grid, we allow a certain degree of freedom in this final fine tuning of the trunk positions.

In all stages of this procedure we refer to trunk positions as trees’ centers of mass. This implies we not only fit trees, but we also aim to fit their positions with maximal margins in all directions.

Algorithm 3 elaborates the detailed trunks approximation and semantic labeling procedure. The algorithms that follow elaborate the implementations of the procedure’s subcomponents. The entire procedure is visualized in a video: [https://youtu.be/7axSA0_6RWA](https://youtu.be/7axSA0_6RWA).

![Figure 22: transformed grid parameters](image)
Algorithm 3: Approximate-And-Label-Trunks

**Input:**
- image [RGB matrix]
- layout [numeric matrix]

**Output:** tree_label_to_trunk_position [dictionary of string $\rightarrow$ pixel coordinates]

**Constants:**
- CROP_RATIO

\[
\begin{align*}
\text{canopies_mask} & := \text{Extract-Canopies-Mask}(\text{image}) \\
\text{cropped_canopies_mask} & := \text{Crop}(\text{canopies_mask}, \text{CROP_RATIO}) \\
\text{orientation} & := \text{Estimate-Tree-Rows-Orientation}(\text{cropped_canopies_mask}) \\
\text{centroids} & := \text{Estimate-Tree-Centroids}(\text{cropped_canopies_mask}, \text{orientation}) \\
\text{aligned_centroids} & := \text{Rotate}(\text{centroids}, \text{orientation}) \\
\text{dim}_x, \text{dim}_y & := \text{Estimate-Trees-Grid-Cell-Dimensions} (\text{aligned_centroids}) \\
\text{shear} & := \text{Estimate-Trees-Grid-Shear}(\text{aligned_centroids}) \\
\text{essential_grid} & := \text{Get-Grid}(\text{dim}_x, \text{dim}_y, \text{orientation}, \text{shear}) \\
\text{origin} & := \text{Find-Grid-Placement}(\text{centroids}, \text{essential_grid}) \\
\text{optimized_grid_args} & := \text{Optimize-Trunks-Grid} (\text{dim}_x, \text{dim}_y, \text{orientation}, \text{shear}, \text{origin}, \text{canopies_mask}) \\
\text{extrapolated_grid} & := \text{Extrapolate-Grid-On-Image}(\text{optimized_grid_args}) \\
\text{plot_origin} & := \text{Align-Layout-On-Extrapolated-Grid}(\text{extrapolated_grid}, \text{optimized_grid_args}[\text{sigma}], \text{layout}, \text{canopies_mask}) \\
\end{align*}
\]

\[
\begin{align*}
\text{m, n} & := \text{Dimensions}(\text{layout}) \\
\text{layout_positions} & := \text{Get-Sub-Matrix}(\text{extrapolated_grid}, \text{plot_origin}, \text{m, n}) \\
\text{trunk_positions} & := \text{layout_positions}[\text{layout} = 1] \\
\text{tree_label_to_trunk_position} & := \text{Label-Trunks-Serially}(\text{trunk_positions}) \\
\text{tree_label_to_trunk_position} & := \text{Refine-Trunk-Positions} (\text{tree_label_to_trunk_position}, \text{canopies_mask}, \text{optimized_grid_args}[\text{dim}_x], \text{optimized_grid_args}[\text{dim}_y], \text{optimized_grid_args}[\text{sigma}]) \\
\end{align*}
\]

For the grid estimation stage, we first crop the central square of the image such that it contains mainly trees from the plot of interest. Figure 23 illustrates the cropped area of the image (inside the blue square). We then extract the canopies mask (Algorithm 2) of the cropped image (Figure 24).
Figure 23: the central cropped region of the image

Figure 24: canopies mask of the cropped image
In order to estimate the tree rows orientation, \( \theta \), we iteratively rotate the cropped canopies mask in the angular range \([-90°, 90°]\), and calculate the column sums vector. The score of each angle is the mean of its column sums vector’s local minima. The chosen orientation is the one that yields the minimal score (Algorithm 4). As the rotated tree rows become perpendicular to the horizontal axis, the column sums vector resembles a sinusoid and thus the mean of minima values is lower. In Figure 25, four exemplifying canopies masks following rotation are illustrated alongside with their column sums vectors.

**Algorithm 4**: Estimate-Tree-Rows-Orientation

---

| Input: canopies_mask [binary matrix] |
| Output: orientation [float number] |
| **Constants**: ANGLE_INCREMENT |

\[
\text{angle_to_score} := \{\}
\]

for angle := -90° to 90° with ANGLE_INCREMENT step size:

\[
\begin{align*}
\text{rotated_canopies_mask} &:= \text{Rotate}(\text{canopies_mask}, \text{angle}) \\
\text{column_sums_vector} &:= \text{Sum-Matrix-Along-Columns}(\text{rotated_canopies_mask}) \\
\text{column_sums_vector_minima} &:= \text{Local-Minima}(\text{column_sums_vector}) \\
\text{score} &:= \text{Mean}(\text{column_sums_vector_minima}) \\
\text{angle_to_score}[\text{angle}] &:= \text{score}
\end{align*}
\]

\[
\text{orientation} := \text{Arg-Min}(\text{angle_to_score})
\]
A: $\theta=0^\circ$
score=127213.12

B: $\theta=2.5^\circ$
(optimal)
score=32056.25

C: $\theta=5^\circ$
score=119696.75

D: $\theta=45^\circ$
score=278939.57

Figure 25: rotated masks with their column sums vectors
In order to estimate the dimensions of a grid cell, $\Delta x$ and $\Delta y$, we first estimate the positions of tree centroids. We rotate the cropped canopies mask by the angle $\theta$ previously found, so that the tree rows are perpendicular to the horizontal axis. The local minima of the column sums vector of the rotated image allows us to determine the horizontal positions of inter-row passage centers (Figure 26). Each tree’s horizontal pose is considered exactly in the middle between its left and right passage centers. We then crop the image to slices according to the passage centers and calculate the row sums vector of each image slice. The local maxima of each row sums vector are the vertical positions of the estimated centroids in that slice (Figure 27). Algorithm 5 elaborates the centroids estimation procedure and Figure 28 illustrates the extracted centroids on the image.

**Algorithm 5:** Estimate-Tree-Centroids

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• canopies_mask [binary matrix]</td>
</tr>
<tr>
<td>• orientation [float number]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>centroids [array of pixel coordinates]</td>
</tr>
</tbody>
</table>

```
rotated_canopies_mask := Rotate(canopies_mask, orientation)
column_sums_vector := Sum-Matrix-Along-Columns(rotated_canopies_mask)
passage_centers := Local-Minima(column_sums_vector)
centroids := {}
for left_passage, right_passage in Adjacent-Pairs(passage_centers):
    canopies_mask_slice := Crop(rotated_canopies_mask, left_passage, right_passage)
slice_row_sums_vector := Sum-Matrix-Along-Rows(canopies_mask_slice)
trees_horizontal_position := Mean(left_passage, right_passage)
trees_vertical_positions := Local-Maxima(slice_row_sums_vector)
    for tree_vertical_position in trees_vertical_positions:
        centroids := centroids U (trees_horizontal_position, tree_vertical_position)
```
Figure 26: column sums minima determine inter-row passage centers
Figure 27: row sums maxima determine centroids’ vertical positions in image slice
Given the estimated tree centroids, we calculate the distances between rows and intra-row distances in order to estimate $\Delta x$ and $\Delta y$. Since not all tree centroids are extracted properly (for example, see the missing centroids in upper part of Figure 28), we rather use percentiles instead of simple mean, as described in Algorithm 6.

**Algorithm 6: Estimate-Trees-Grid-Cell-Dimensions**

<table>
<thead>
<tr>
<th>Input:</th>
<th>centroids [array of pixel coordinates]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>dim_x [float number]</td>
</tr>
<tr>
<td></td>
<td>dim_y [float number]</td>
</tr>
<tr>
<td>Constants:</td>
<td>Q_PERCENTILE_X</td>
</tr>
<tr>
<td></td>
<td>Q_PERCENTILE_Y</td>
</tr>
</tbody>
</table>

```plaintext
horizontal_distances := Between-Rows-Euclidean-Distances(centroids) vertical_distances := Intra-Row-Euclidean-Distances(centroids) dim_x := Percentile(horizontal_distances, Q_PERCENTILE_X) dim_y := Percentile(vertical_distances, Q_PERCENTILE_Y)
```
To complete the estimation of an initial grid, we estimate the shear $\alpha$ in Algorithm 7. We first find “drift vectors”, i.e., the vectors from each centroid to the closest centroid in the row to the right (Figure 29). We calculate the angle of these vectors relative to the horizontal axis of the rotated image and reject extreme angle values which we consider outliers (Figure 30). Again, we prefer using percentile to calculate the estimated shear.

**Algorithm 7:** Estimate-Trees-Grid-Shear

| Input: centroids [array of pixel coordinates] |
| Output: shear [float number] |
| **Constants:** Q_PERCENTILE |

```plaintext
angles := {}
for centroid in centroids:
    closest_centroid := Find-Closest-Point-In-Right-Row(centroid, centroids)
    if closest is not N/A:
        angle := Vector-Angle(centroid, closest_centroid)
        angles := angles U angle
    angles := Remove-Extreme-Values(angles)
    shear := Percentile(angles, Q_PERCENTILE)
```

![Figure 29: “drift vectors” between centroids](image-url)
At this point of the process, we have a rough estimation of the initial grid parameters $\Delta x, \Delta y, \theta$ and $\alpha$. We now want to place this grid on the image and find its origin $(O_x, O_y)$. For this purpose, we build an initial 6×6 grid using the four estimated parameters mentioned above. In addition, we use the estimated tree centroids previously found (using Algorithm 5). We place the grid’s origin vertex on one estimated centroid such that the sum of Euclidean distances between the grid’s vertices and their matching closest centroids is minimal (Algorithm 8). Figure 31 illustrates the grid placement with minimal sum of Euclidean distances. The vectors from each grid vertex (blue point) to its closest centroid (red point) are marked with cyan lines.
**Algorithm 8**: Find-Grid-Placement

**Input:**
- centroids [array of pixel coordinates]
- grid [array of pixel coordinates]

**Output**: origin [pixel coordinates]

```plaintext
origin_to_score := {}
for candidate_origin in centroids:
    placed_grid := Place-Grid-In-Origin(grid, candidate_origin)
    score := 0
    for vertex in placed_grid:
        closest_centroid := Find-Closest-Point(vertex, centroids)
        score := score + Euclidean-Distance(vertex, closest_centroid)
    origin_to_score[candidate_origin] := score
origin := Arg-Min(origin_to_score)
```

Figure 31: optimal placement of the 6×6 grid (blue points) on the estimated centroids (red points)
As evident in Figure 31, the initial estimation of the 6×6 grid is suboptimal. Therefore, we chose to fine-tune this grid to better fit the image in its local vicinity. The fine-tuning task is addressed as an optimization problem whose initial guess is the grid estimated above and whose objective function reflects the fitting of the grid to the image. The image representation remains the binary canopies mask previously extracted, but with a slight change: black zero-valued pixels are converted to -1. The newly created matrix is practically a reward mask that “encourages” canopy pixels and “penalizes” non-canopy pixels. Bearing in mind that a canopy is a radial object but certainly not a perfect circle, we chose to represent the tree grid as a grid of cropped gaussians (Figure 32). The center of a gaussian represents its approximated trunk positions on the image. The pixelwise multiplication of the reward mask with the gaussians yields a matrix whose values range in [-1,1]. All non-zero values are pixels that fall under a gaussian; the positive ones match canopy pixels (Figure 33) and the negative ones match non-canopy pixels. Zero values in the matrix match pixels which are not under a gaussian and therefore get neither positive nor negative reward. The summation of pixels in the matrix around grid vertices yields a tree score per vertex (Algorithm 9). Providing that tree scores are closely distributed, their mean value reflects an overall fitting score of the entire 6×6 grid to the image. In Algorithm 10 we use this mean value to create the target function that we want to maximize.
Algorithm 9: Calculate-Tree-Score

Input:
- point [pixel coordinates]
- sigma [float number]
- canopies_mask [binary matrix]

Output: score [float number]

reward_mask := Replace-Value-In-Matrix(canopies_mask, 0, -1)
gaussian_mask := Get-Gaussian-Mask(point, sigma)
score := Sum(Pixelwise-Multiply(reward_mask, gaussian_mask))

Algorithm 10: Calculate-Grid-Score

Input:
- dim_x (float number)
- dim_y (float number)
- orientation (float number)
- shear (float number)
- origin (pixel coordinates)
- sigma (float number)
- canopies_mask [binary matrix]

Output: score [float number]

Constants: TREE_SCORES_STD_THRESHOLD

tree_scores := {}  
grid := Get-NxN-Gaussians-Grid(dim_x, dim_y, orientation, shear, origin, sigma, 6)  
for vertex in grid:  
  tree_score := Calculate-Tree-Score(vertex, sigma, canopies_mask)  
  tree_scores := scores ∪ tree_score  
normalized_tree_scores := Elementwise(Normalize-By-Gaussian-Volume(tree_scores, sigma))  
if Standard-Diviation(normalized_tree_scores) > TREE_SCORES_STD_THRESHOLD:  
  score := −\infty  
else:  
  score := Mean(scores)
The optimization variables need to control the gaussian centers as well as their widths. As mentioned above, the 6×6 grid is well defined by six parameters: the grid dimensions $\Delta x$ and $\Delta y$, the orientation angle $\theta$, the shear angle $\alpha$ and the grid’s placement on the image $(O_x,O_y)$. Gaussian widths also need to be controlled by the optimization solver such that wider gaussians, which cover more canopy area, are preferred and the gaussian centers are consequently better aligned with canopy centroids. To avoid high computational complexity, we assume that canopy sizes are nearly equal, and thus, all gaussian widths are parametrized using a single $\sigma$ variable, denoting their standard deviation in pixels. We chose to work with a Nelder-Mead solver [37] which does not require any gradient calculations, as those might be computationally expensive with seven variables. In addition, we bounded the search range in each of the parameters around the initial estimated values (Algorithm 11). Figure 34 illustrates the gaussians grid alongside with their matching variable values and score in selected stages of the optimization process. Figure 35 demonstrates the discrepancy between the initial grid and the output of the optimization process. In this example and in many others, it is clear that the Nelder-Mead optimization improved the approximated trunk positions.

**Algorithm 11: Optimize-Trunks-Grid**

**Input:**
- init_dim_x (float number)
- init_dim_y (float number)
- init_orientation (float number)
- init_shear (float number)
- init_origin (pixel coordinates)
- canopies_mask [binary matrix]

**Output:** optimized_grid_args [struct of variable → value]

**Constants:**
- GRID_DIMS_MARGIN
- ORIENTATION_MARGIN
- SHEAR_MARGIN
- TRANSLATION_MARGIN
- SIGMA_MARGIN
- INITIAL_SIGMA_TO_DIM_Y_RATIO

init_sigma := init_dim_y × INITIAL_SIGMA_TO_DIM_Y_RATIO
optimized_grid_args := Nelder-Mead-Maximization(target := Calculate-Grid-Score,

initial_values := (init_dim_x, init_dim_y,
init_orientation, init_shear,
init_origin, init_sigma),
ranges := (GRID_DIMS_MARGIN,
GRID_DIMS_MARGIN,
ORIENTATION_MARGIN,
SHEAR_MARGIN,
TRANSLATION_MARGIN,
SIGMA_MARGIN),
arguments := canopies_mask)
A: step #1

$\Delta x = 283$
$\Delta y = 209$
$\theta = 2.5$
$\alpha = 9.3$
$O_x = 655$
$O_y = 1197$
$\sigma = 68.84$

score = 2757564.42

B: step #31

$\Delta x = 273$
$\Delta y = 221$
$\theta = 2.44$
$\alpha = 9.07$
$O_x = 657$
$O_y = 1199$
$\sigma = 68.21$

score = 3905645.61

C: step #52 (optimal)

$\Delta x = 271$
$\Delta y = 227$
$\theta = 3.4$
$\alpha = 8.89$
$O_x = 658$
$O_y = 1202$
$\sigma = 84.22$

score = 4732202.62

Figure 34: selected stages in the Nelder-Mead optimization process
With the 6×6 grid estimation in hand, we next align the plot pattern described by the farmer’s layout on the image. To do this, we extrapolate the optimized 6×6 grid on the entire image (Figure 36). Then, in Algorithm 12 we calculate the tree score of each vertex in the extrapolated grid using Algorithm 9 and normalize by the gaussian volume (Figure 37). We arrive at a score in [-1, 1] for every vertex where 1 denotes complete green area and -1 denotes complete non-green area. In the following stage, we loop over all possible placements of the layout matrix on the extrapolated grid and calculate the elementwise multiplication between the layout matrix and the scores of the relevant sub-grid. Providing that tree scores range in [-1, 1] and the layout’s allowed values are {1, -1, 0}, we get a match score per vertex in [-1, 1] where 1 denotes compete match, -1 denotes complete mismatch and 0 denotes a neutral vertex. The mean value of this matrix provides a total match score which we use to choose the optimal placement. Figure 38 demonstrates two possible placements of the layout with the vertices’ match scores.
Figure 36: the optimized grid extrapolated on the entire image

Algorithm 12: Align-Layout-On-Extrapolated-Grid

Input:
- extrapolated_grid [array of pixel coordinates]
- optimized_sigma [float number]
- layout [numeric matrix]
- canopies_mask [binary matrix]

Output: plot_origin [pixel coordinates]

vertex_to_tree_score := {}
for vertex in extrapolated_grid:
    tree_score := Calculate-Tree-Score(vertex, optimized_sigma, canopies_mask)
    normalized_tree_score := Normalize-By-Gaussian-Volume(tree_score, optimized_sigma)
    vertex_to_tree_score[vertex] := normalized_tree_score
m, n := Dimensions(layout)
origin_to_match_score := {}
for sub_grid_origin in extrapolated_grid:
    sub_grid_scores := Get-Sub-Grid-Scores-Matrix(vertex_to_tree_score, sub_grid_origin, m, n)
    layout_match_score := Mean(Elementwise-Multiply(sub_grid_scores, layout))
    origin_to_match_score[sub_grid_origin] := layout_match_score
plot_origin := Arg-Max(origin_to_match_score)
Figure 37: normalized tree scores of the extrapolated grid vertices
**A: suboptimal placement (mean score is 0.45)**

**B: optimal placement (mean score is 0.8)**

Figure 38: optimal vs sub-optimal placement of the layout
Having the layout matched to the grid, we can now label the trees. Each label is comprised of a number and a letter, the first denoting the ordinal row number and the latter denoting the tree’s ordinal number in that row (Figure 39).

Providing that orchards trees do not perfectly match a grid pattern, we allow for final refinement of trunk positions. The refinement procedure receives the trunk positions estimated previously and iteratively refines each trunk individually by slightly moving its pose and searching for the maximal tree score (Algorithm 13). The ultimate result of the trunks approximation and labeling procedure is Figure 19. Additional examples are provided in appendix 9.2. Our ability to reproduce successful approximation of trunk positions in multiple images from different times of the day and different times of the year is an indication of the robustness of this procedure.

Figure 39: trees layout matched to the image with semantic labeling
**Algorithm 13:** Refine-Trunk-Positions

**Input:**
- trunk_positions [array of pixel coordinates]
- canopies_mask [binary matrix]
- dim_x [float number]
- dim_y [float number]
- sigma [float number]

**Output:** refined_trunk_positions [array of pixel coordinates]

**Constants:**
- WINDOW_EXPANSION
- SIGMA_SCALE

```
window_size := Max(dim_x, dim_y) × WINDOW_EXPANSION
search_range := sigma × SIGMA_SCALE
refined_trunk_positions := {}
for original_x, original_y in trunk_positions:
    position_to_score := {}
    for x := (original_x – search_range) to (original_x + search_range):
        for y := (original_y – search_range) to (original_y + search_range):
            window := Crop-Image-Around(canopies_mask, (x, y), window_size)
            score := Calculate-Tree-Score((x, y), optimized_sigma, window)
            position_to_score[x, y] := score
    refined_trunk_position := Arg-Max(position_to_score)
    refined_trunk_positions := refined_trunk_positions U refined_trunk_position
```
4.2.1 Image Alignment

A direct by-product of trunks approximation is the ability to align images by the trunk positions. Alignment of the top-view images can be useful in any application that involves multiple observations of the orchard throughout time.

We estimate the affine transformation between two orchard images by first applying the trunks approximation procedure on both. We then estimate the affine transformation [38] between the two trunk point sets and use the estimated transformation to warp one image on another. Figure 40 demonstrates the alignment of two images acquired in the April experiment.

In order to evaluate the suggested image alignment approach, we compare it against a typical image alignment flow which includes the extraction of ORB features [39] from the two images. The two extracted point sets are then used to estimate the affine transformation using the same method mentioned above. In addition, we use the positions of the four bright blue markers as ground truth points in order to estimate the actual affine transformation. The similarity between two aligned images is quantified by two common metrics for image similarity: Mean Square Error (MSE) and Structural Similarity Index Measure (SSIM). For convenience, we calculate $1 - \text{MSE}$ as a measure of similarity. Both measures of similarity range in [0,1] such that 1 denotes perfect match and 0 denotes perfect mismatch. We calculate the above metrics on a center crop of the canopies masks following the alignment. Table 14 summarize the similarity results of selected images in the April and November experiments and in two different orchard plots. The table columns (“by trunks”, “by ORB”, “by markers”) contain results of the above mentioned three alignment methods. The colored bars visualize the cells’ magnitude.
Figure 40: image alignment using approximated trunk positions
Table 14: similarity of selected pairs of images

<table>
<thead>
<tr>
<th>plot #</th>
<th>image to align</th>
<th>baseline image</th>
<th>1-MSE by trunks</th>
<th>1-MSE by ORB</th>
<th>1-MSE by markers</th>
<th>SSIM by trunks</th>
<th>SSIM by ORB</th>
<th>SSIM by markers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>April, 15:08</td>
<td>April, 15:53</td>
<td>0.928856009</td>
<td>0.9055803</td>
<td>0.913561894</td>
<td>0.824765478</td>
<td>0.800899221</td>
<td>0.804706791</td>
</tr>
<tr>
<td>1</td>
<td>April, 15:08</td>
<td>April, 16:55</td>
<td>0.909254314</td>
<td>0.907248735</td>
<td>0.845332569</td>
<td>0.782361629</td>
<td>0.781022369</td>
<td>0.712537885</td>
</tr>
<tr>
<td>1</td>
<td>April, 15:08</td>
<td>April, 19:04</td>
<td>0.91516924</td>
<td>0.415619729</td>
<td>0.826943592</td>
<td>0.797260264</td>
<td>0.351522847</td>
<td>0.701040768</td>
</tr>
<tr>
<td>1</td>
<td>April, 15:53</td>
<td>April, 16:55</td>
<td>0.919639897</td>
<td>0.495123002</td>
<td>0.879826544</td>
<td>0.794389509</td>
<td>0.37314978</td>
<td>0.748305442</td>
</tr>
<tr>
<td>1</td>
<td>April, 15:53</td>
<td>April, 19:04</td>
<td>0.888315566</td>
<td>0.645007136</td>
<td>0.834101868</td>
<td>0.76567459</td>
<td>0.519383432</td>
<td>0.706861586</td>
</tr>
<tr>
<td>1</td>
<td>April, 16:55</td>
<td>April, 19:04</td>
<td>0.827245419</td>
<td>0.592190856</td>
<td>0.791765774</td>
<td>0.700336919</td>
<td>0.47393258</td>
<td>0.65384269</td>
</tr>
<tr>
<td>1</td>
<td>November, 10:09</td>
<td>November, 10:10</td>
<td>0.814501975</td>
<td>0.886031585</td>
<td>0.883488121</td>
<td>0.61526862</td>
<td>0.703936394</td>
<td>0.696364016</td>
</tr>
<tr>
<td>2</td>
<td>November, 11:07-1</td>
<td>November, 11:07-2</td>
<td>0.846647097</td>
<td>0.890641893</td>
<td>0.859913111</td>
<td>0.644672151</td>
<td>0.705759537</td>
<td>0.658281055</td>
</tr>
</tbody>
</table>

The “by markers” results in Table 14, which should theoretically be close to 1, are an indication for the metrics’ inability to fully express image similarity in our case. However, they do provide reference numbers for a good alignment. In most cases demonstrated in the tables, “by trunks”, “by ORB” and “by markers” have similar scores (“by trunks” is 10% higher than “by ORB” on average). Nevertheless, in certain cases, the ORB method leads to fatal failure of the alignment (Figure 41) and therefore we consider it less robust compared with the trunks method.

Figure 41: bad alignment of 15:08 to 19:04 (April) using the typical ORB method
5 Navigation Architectures

The computer vision techniques presented above serve us to derive insights from the top view images that can improve various navigation aspects of the ground vehicle.

Two main considerations are to be considered for acquisition of top-view images:

- **Image frequency**: for tasks requiring continuous monitoring, a camera must be constantly recording a video. In such tasks, the video is typically processed in real-time. In other tasks, periodic snapshot images acquired opportunistically are sufficient, and the need for continuous recording is eliminated.

- **Camera altitude**: the required field of view varies from one task to another. For certain tasks, high altitude images which capture a larger area at the ground level are required, while in other tasks only a smaller area is sufficient.

Satellite imagery is long known to serve farmers for various purposes, for example in disease identification [40]. Despite impressive resolution of modern satellite imagery, they still do not provide centimeter-level resolution. More importantly, satellite image frequency do not allow neither continuous monitoring nor even minute-level monitoring of orchards. Hence, for this work satellite images are not suitable.

With the technology advances of recent years, UAVs are becoming more robust and affordable and hence more common in agriculture. Additionally, when a UAV in an orchard is elevated above tree canopies level, it does not suffer from GPS signal occlusions as at ground level, and hence is able to better position and stabilize itself. For this work, we use a UAV that hovers above the orchard and is equipped with a downward looking camera. For image acquisition purposes, UAVs allow flexibility in terms of both image frequency and camera altitude.

Despite the clear advantages of UAVs, their technical limitations are ample. First and foremost, UAVs are limited in their work time. With state-of-the-art technologies, batteries typically allow flight time of several minutes only. This implies difficulties in tasks which require continuous monitoring. Second, flight at high altitude occasionally results in bad connectivity of the UAV to the ground. This might be problematic in real-time applications, especially when the data is streamed from the UAV and processed remotely on the ground. Additional limitations of UAVs relate to flight regulations. Specifically in agriculture, UAV flying at high altitude might interfere with crop dusting aerial activities.

With the understanding of the above limitations and considering that farmers work under different restrictions and have different needs, we are suggesting two groups of navigation architectures which can either be used exclusively or in conjunction:

- **Low altitude**: the UAV hovers at relatively low altitudes, tracks and follows the UGV to provide a video stream of the tracked UGV at close range. These low altitude solutions are designed to work continuously and in real-time. A possible solution to bypass the problem of limited battery time is to use tethered UAVs (e.g., [41]), powered by higher capacity batteries on the ground (possibly on the UGV). In theory, a downward looking camera can also be placed at the top of a mast mounted on the UGV, however these may cause severe stability problems. Figure 10 is one frame of a video that we recorded in the field with our UAV at an altitude of 20 meters.
• **High altitude:** the UAV periodically elevates to a point from which it can capture one snapshot image of an entire orchard plot. In this group of architectures, the image is processed and compared against previous images of that plot. This approach allows one UAV to serve, in turns, on many plots of one orchard. In our experiments we captured images at altitude of 60 to 80 meters (Figure 9).

5.1 **LOW ALTITUDE**

The innovative approach that we suggest for low altitude tasks is connected to canopy contours. In low altitudes, top-view images capture a small portion of the orchard in high detail, leading to high quality canopy extractions (using Algorithm 2).

Common localization solutions use two-dimensional LiDAR scans as their main sensor input. These scans are used for understanding local and incremental translation and rotation changes (e.g., ICP [27]), but also for global localization by matching the scans against a prior map, and by that addressing the kidnapped robot problem (e.g., AMCL [22]). Motivated by the extensive use of LiDAR scans for localization and mapping of ground vehicles, we decided to generate a “virtual canopies scan” which depicts the shape of the canopy contours. As a LiDAR scan, the virtual canopies scan measures distances from an origin point where a robot is located to its surroundings. The surroundings in this case are the canopies. The virtual scan is calculated using computer vision techniques from the top-view image. Algorithm 15 describes the virtual scan generation procedure (Appendix 9.3 elaborates the calculation of the pixel-to-meter ratio) and Figure 42 illustrates one virtual canopies scan on the orchard image (the blue circle denotes the maximal scan range). The canopies scans form a heterogenous representation of the orchard, and as already stated, this heterogeneity is vital for scan-based algorithms. These scans are also advantageous for being a compact representation that is also compatible with existing localization and mapping software libraries.

The video in [https://youtu.be/7lm4hED9ck0](https://youtu.be/7lm4hED9ck0) illustrates the tracking of a UGV and the extraction of virtual canopies scans in a real video acquired from a UAV at 20 meters.
Algorithm 15: Generate-Virtual-Canopies-Scan

**Input:**
- canopies_mask [binary matrix]
- origin [pixel coordinates]
- pixel_to_meter_ratio [float number]

**Output:** scan [array of float numbers]

**Constants:**
- NUMBER_OF_SAMPLES
- MIN_RADIUS
- MAX_RADIUS
- SEARCH_DELTA

```plaintext
scan[0:NUMBER_OF_SAMPLES] := {NULL}
for sample_index := 0 to NUMBER_OF_SAMPLES:
    angle := 360° / sample_index
    for radius := MIN_RADIUS to MAX_RADIUS with SEARCH_DELTA step size:
        pixel := origin + (radius × Cos(angle), radius × Sin(angle))
        if canopies_mask[pixel] = 1:
            scan[sample_index] := radius × pixel_to_meter_ratio
            break
```

Figure 42: virtual canopies scan (red) around an origin point (purple)
5.1.1 Canopy-Based AMCL

As already stated in chapter 2, MCL is a localization approach which can address the kidnapped robot problem. MCL assumes a heterogeneous environment and therefore could not have been used thus far by robots using two-dimensional LiDARs in orchards. The canopy contours however, do provide a heterogeneous observation of the orchard. This provides an opportunity for the virtual canopies scans to help addressing the localization problem in orchards.

The concept of this architecture comprises two stages. First, a reference map of the orchard is required. One high altitude image that captures the entire plot is sufficient. Using the acquired image, the canopies are extracted, and a map is generated (Figure 43). Second, a continuous top-view video stream records the UGV in the orchard from low altitude (e.g., using a tracking UAV). For each frame of the stream, the vehicle is segmented from the image and its location is used as the origin point for generating the canopies scan (Algorithm 15). Assuming that the orchard ground is nearly planar, the vehicle’s state is a two-dimensional vector \((x, y)\), which is relative to the map origin. MCL operates on the canopies scans and estimates the vehicle’s two-dimensional pose against the canopy map.

![canopy map](image)

Figure 43: canopy map
We chose to use ROS’s implementation of AMCL [42] whose adaptive resampling typically results in lower computational complexity and faster convergence. This implementation estimates the three-dimensional \((x, y, \theta)\) state of a ground robot. To avoid unnecessary complexity and potential divergence following the estimation of \(\theta\), we modified the source code of AMCL such that its initial particles are bounded in a narrow angular range. AMCL’s motion update stage relies on an odometry model which accounts for noises and drifts. We chose the most generic omnidirectional model which can describe any robot as a point mass which can translate in all directions (see motion model parameter values in appendix 9.4). Figure 44 illustrates the convergence of AMCL particles on the canopies map.

In order to evaluate this approach and demonstrate its robustness, we conducted numerous synthesized experiments using the data collected in the field. In each experiment, we generate a virtual UGV trajectory in the orchard using the approach suggested in 5.2.2 and using the approximated trunk positions as waypoints. Figure 45 illustrates the different trajectories that we examined. Trajectories A and B are basic patterns along one passage, C and D are S-shaped and fork-shaped patrol patterns, and E is a trajectory combining a series of tasks in various points in the orchard. Each experiment uses one high altitude image as a map and simulates the low altitude video stream by cropping small windows of another high altitude image along the calculated trajectory. The use of two different images acquired at different times of the day simulates the actual mode of work suggested in this architecture. The virtual trajectories are advantageous for providing the ground truth pose of the virtual vehicle which will serve us for evaluation purposes. As an odometry source for AMCL, we use the incremental translation between adjacent points of the trajectory.
Figure 44: the gradual convergence process of AMCL particles
Figure 45: virtual trajectories (plot #1)
In order to better evaluate our approach, we want to not only measure our performance but also compare it against a reference. As mentioned above, two-dimensional LiDARs are typically mounted on the UGV such that they acquire a horizontal laser scan at the trunks level. To mimic the typical LiDAR mode of operation, we simulate trunks images and apply the same AMCL-based approach elaborated above for the canopies. The trunks images are generated based on the approximated trunk positions in the images used in that experiment; trunk positions outside the relevant plot were marked manually. Since trunks are not perfect circles, we place ellipses centered at the approximated trunk positions with a certain degree of randomness in their axis lengths and orientation angle. Figure 46 illustrates the trunk map aligned with the canopy map as well as the canopies and trunks AMCL instances.

The trunks AMCL instance mitigated conditions when comparing with real-world LiDAR scans. First, the angular range of the trunks scans is 360° whereas typical real-world LiDARs provide lower ranges. Second, we ignore the effect of terrain bumps and provide a continuous smooth scan which constantly perceives the surrounding trunks. Moreover, while the canopies scans suffer from discontinuities while the UGV passes beneath a canopy, the trunks scans are continuous and valid throughout the entire trajectory. In the following experiments we will show that the canopies instances demonstrate better performance despite the above-mentioned advantages which the trunks instances have over them.

The video in https://youtu.be/_pWT8LPzixA visualizes one execution of the two AMCL instances along one trajectory.

Figure 46: canopies (green) vs trunks (brown): AMCL particles and aligned maps
For each of the two AMCL instances, we extract two output temporal signals. Most trivially, we are interested in the temporal pose error, calculated as the Euclidean distance between the UGV’s pose (estimated by AMCL) and the ground truth pose, i.e., the pose on the virtual trajectory. The pose error is served as a measure of accuracy. An additional output of AMCL is a pose covariance matrix. We calculate the Frobenius norm of this matrix at every timestamp and that is used as a measure of certainty of AMCL. Since AMCL is stochastic by its nature, we repeat every experiment ten times. At every timestamp we calculate the standard deviation of the ten repetitions, which reflects the degree of nondeterminism.

Figure 47 illustrates the average pose error and covariance norm against the time in five experiments. The averaging is over the ten repetitions of each experiment. The five experiments are five different trajectories (Figure 45) and they were conducted on the same pair of high altitude images, one used as a map the other used to simulate the low altitude tracking videos. The standard deviation of both signals is also visualized on the graphs as partially transparent areas. In most of the cases in the figure, it is evident that the canopies AMCL instance converges faster to the true pose, with better certainty and with lower variance between the repetitions, which implies better robustness. As seen in some of the trajectories in the figure, the canopies curves can rise for short periods of time following the first convergence. This behavior stems from trajectory sections which fall beneath a canopy and therefore no scans are available during these short periods.
Figure 47: average AMCL pose error and covariance norm against the time
(map image: April, 15:08; tracking image: April, 15:53)
In order to compare between trunks and canopies, we define five quantitative measures of an experiment which aggregate the results of all ten repetitions. As our interest is in centimeter-level accuracy, we define a one-meter error band for the pose errors. A repetition of an experiment is said to converge if the pose error curve enters the one-meter error band (Figure 48).

Convergence rate is defined as the portion of experiment repetitions which converged among the ten repetitions:

\[
\text{CONVERGENCE RATE} = \frac{\# \text{ converged repetitions}}{10}
\]

The second measure relates to the settling time of the pose error. We define settling time of a single repetition \( t_s^{(i)} \) as the first time the pose error curve enters the error band (Figure 48). Even though AMCL pose error might exceed the one-meter band following its first entrance, this time reflects AMCL’s speed of convergence. To aggregate all repetitions, we calculate the average settling time of the converged repetitions:

\[
\text{SETTLING TIME} = \frac{\sum_{i \in \text{converged repetitions}} t_s^{(i)}}{\# \text{ converged repetitions}}
\]

The third measure expresses the portion of time in which the pose error is within the error band, i.e., the portion of time with centimeter-level localization accuracy. For a single repetition, this quotient is calculated as \( t_{IB}^{(i)} = \sum_j \Delta t_{IB,j}/T \) (Figure 48). This measure practically quantifies the relative amount of time with centimeter-level accuracy. Again, we aggregate all repetition results by averaging:

\[
\text{IN-BAND RATE} = \frac{\sum_i t_{IB}^{(i)}}{10}
\]

The fourth measure of an experiment quantifies AMCL’s uncertainty. For each repetition, we calculate the mean value of covariance norms over the time \( E[\|\text{Cov}\|]^{(i)} \). Lower uncertainty values indicate on better confidence of the algorithm in its estimation. Again, we average over all repetitions:

\[
\text{UNCERTAINTY} = \frac{\sum_i E[\|\text{Cov}\|]^{(i)}}{10}
\]

The fifth measure quantifies the variance between the ten repetitions of an experiment and reflects the degree of determinism. Low variance implies high degree of determinism and better robustness. At each timestamp, we calculate the variance of pose errors over the ten repetitions \( \sigma(t) \) (Figure 49). As a total variance we take the mean value of \( \sigma(t) \) over the time:

\[
\text{VARIANCE} = E[\sigma(t)]
\]
Figure 48: figurative graph of the pose error against the time in a single experiment repetition

Figure 49: figurative graph of the average pose error and its standard deviation against the time

Table 16 below depicts the aggregated results of all thirty experiments conducted on the April data (the colored bars visualize the cells’ magnitude). For each combination of two images among the four collected throughout the day (Figure 19, Figure 62, Figure 63 and Figure 64) that are used as map and tracking sources, the five trajectories mentioned in Figure 45 are calculated using the same semantic waypoints and the procedure described in chapter 5.2.2. As already mentioned, each experiment consisted of ten repetitions. The measures in the table show clear advantage for the canopies instances, despite the above mentioned mitigated conditions of the trunks. In terms of convergence rate, in-band rate and settling time, the canopies instance is almost always outperforming. The uncertainty of the canopies is lower in most cases compared with the trunks, implying stronger confidence of the algorithm in its estimated pose. Another interesting observation is the variance which is by magnitudes lower for the canopies instances in most cases. The canopies instances fail in the “narrow passage” trajectories in certain cases. This is explained by the lower number of valid scans in these trajectories (around 70% only).
The five trajectories are presented in Figure 50 and the results are in Table 18.

Table 17 depicts the results of another five experiments conducted on images acquired in November at the same plot used in the April experiment (Figure 65 and Figure 66) and using the same trajectory waypoints. Again, the canopies demonstrate better performance in most cases. The “narrow passage” trajectory still suffers from the lower number of available scans which leads to AMCL failure.

We ran five experiments also on images acquired in the other plot in November (Figure 67 and Figure 68). The five trajectories are presented in Figure 50 and the results are in Table 18.
A: narrow passage
B: wide passage
C: S-shaped patrol
D: fork-shaped patrol
E: series of tasks

Figure 50: virtual trajectories (plot #2)
Table 18: aggregated AMCL results (November, plot #2)

<table>
<thead>
<tr>
<th>map image</th>
<th>tracking image</th>
<th>trajectory</th>
<th>convergence rate canopies</th>
<th>trunks</th>
<th>in-band rate canopies</th>
<th>trunks</th>
<th>settling time canopies</th>
<th>trunks</th>
<th>uncertainty canopies</th>
<th>trunks</th>
<th>variance canopies</th>
<th>trunks</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:07-1</td>
<td>11:07-2</td>
<td>narrow passage</td>
<td>0.00</td>
<td>0.3</td>
<td>0.00</td>
<td>0.01</td>
<td>8.66</td>
<td>209.22</td>
<td>559.86</td>
<td>3.44</td>
<td>18.48</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>wide passage</td>
<td>1.00</td>
<td>0.0</td>
<td>0.10</td>
<td>0.00</td>
<td>60.02</td>
<td>111.48</td>
<td>420.39</td>
<td>4.19</td>
<td>9.59</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>S-shaped patrol</td>
<td>1.00</td>
<td>0.6</td>
<td>0.11</td>
<td>0.17</td>
<td>52.07</td>
<td>51.95</td>
<td>83.60</td>
<td>1.25</td>
<td>12.27</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fork-shaped patrol</td>
<td>1.00</td>
<td>0.9</td>
<td>0.86</td>
<td>0.03</td>
<td>28.90</td>
<td>34.18</td>
<td>33.81</td>
<td>131.55</td>
<td>0.81</td>
<td>12.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>series of tasks</td>
<td>0.70</td>
<td>0.8</td>
<td>0.01</td>
<td>0.16</td>
<td>91.26</td>
<td>60.18</td>
<td>110.78</td>
<td>128.13</td>
<td>5.35</td>
<td>9.35</td>
</tr>
</tbody>
</table>

As in the previous sets of experiments, the canopies demonstrated overall better performance and they fail mainly in the “narrow passage” experiment in which there are no available scans along significant parts of the trajectory.
5.1.1.1 Canopy-based AMCL with Synthesized Noise

To further stress the suggested canopy-based AMCL approach, we conducted a series of experiments with various types of synthesized noise. We selected eight among the experiments in Table 16 in which the performance of the trunks AMCL instance was competitive with the canopies instance and reran each of them multiple times with the various noises.

The first type of noise that we synthesized is measurement noise. We simply added to the virtual scans an Independent and Identically Distributed (IID) gaussian noise with zero mean $N(0, \sigma^2)$. In reality, this type of noise can account for inaccuracies in the canopy extraction procedure but also for slight variations in canopy shape, e.g., due to windy weather. We modified the $\sigma$ parameter in the range $[0.1, 0.5]$ meters. Figure 51 illustrates one noisy scan with $\sigma = 0.5$.

Table 19 concentrates the results of all experiments with measurement noise ($\sigma = 0$ is also listed as a reference). Despite some deterioration in the canopies instances as $\sigma$ increases, the measures show reasonable performance in most cases. In the trunks instances on the other hand, the degree of performance deterioration is more significant, indicating on lower robustness to this type of noise.

![Figure 51: synthesized measurement noise](image-url)
<table>
<thead>
<tr>
<th>map image</th>
<th>tracking image</th>
<th>trajectory</th>
<th>sigma</th>
<th>convergence rate</th>
<th>in-band rate</th>
<th>settling time</th>
<th>uncertainty</th>
<th>variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:08</td>
<td>15:53</td>
<td>wide passage</td>
<td>0.1</td>
<td>0.9</td>
<td>0.52</td>
<td>0.21</td>
<td>28.40</td>
<td>45.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>1.0</td>
<td>0.66</td>
<td>0.37</td>
<td>31.50</td>
<td>53.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td>0.9</td>
<td>0.67</td>
<td>0.44</td>
<td>42.34</td>
<td>55.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td>0.7</td>
<td>0.71</td>
<td>0.40</td>
<td>53.23</td>
<td>55.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td>0.7</td>
<td>0.71</td>
<td>0.40</td>
<td>53.23</td>
<td>55.43</td>
</tr>
<tr>
<td>15:08</td>
<td>15:53</td>
<td>S-shaped patrol</td>
<td>0.1</td>
<td>0.7</td>
<td>0.67</td>
<td>0.21</td>
<td>56.83</td>
<td>90.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>0.6</td>
<td>0.77</td>
<td>0.42</td>
<td>63.85</td>
<td>97.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td>0.6</td>
<td>0.71</td>
<td>0.40</td>
<td>72.87</td>
<td>72.87</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td>0.6</td>
<td>0.71</td>
<td>0.40</td>
<td>72.87</td>
<td>72.87</td>
</tr>
<tr>
<td>15:53</td>
<td>16:55</td>
<td>wide passage</td>
<td>0.1</td>
<td>1.0</td>
<td>0.92</td>
<td>0.74</td>
<td>32.72</td>
<td>52.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>0.8</td>
<td>0.82</td>
<td>0.74</td>
<td>52.17</td>
<td>52.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td>0.7</td>
<td>0.71</td>
<td>0.40</td>
<td>63.85</td>
<td>63.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td>0.7</td>
<td>0.71</td>
<td>0.40</td>
<td>72.87</td>
<td>72.87</td>
</tr>
<tr>
<td>15:53</td>
<td>16:55</td>
<td>fork-shaped patrol</td>
<td>0.1</td>
<td>1.0</td>
<td>0.92</td>
<td>0.74</td>
<td>32.72</td>
<td>52.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>0.8</td>
<td>0.82</td>
<td>0.74</td>
<td>52.17</td>
<td>52.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td>0.7</td>
<td>0.71</td>
<td>0.40</td>
<td>63.85</td>
<td>63.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td>0.7</td>
<td>0.71</td>
<td>0.40</td>
<td>72.87</td>
<td>72.87</td>
</tr>
<tr>
<td>15:53</td>
<td>19:04</td>
<td>fork-shaped patrol</td>
<td>0.1</td>
<td>0.9</td>
<td>0.56</td>
<td>0.40</td>
<td>26.77</td>
<td>41.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>0.9</td>
<td>0.40</td>
<td>0.56</td>
<td>26.77</td>
<td>41.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td>0.9</td>
<td>0.56</td>
<td>0.40</td>
<td>26.77</td>
<td>41.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td>0.9</td>
<td>0.56</td>
<td>0.40</td>
<td>26.77</td>
<td>41.89</td>
</tr>
<tr>
<td>16:55</td>
<td>19:04</td>
<td>narrow passage</td>
<td>0.1</td>
<td>1.0</td>
<td>0.77</td>
<td>0.42</td>
<td>17.92</td>
<td>57.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>0.8</td>
<td>0.77</td>
<td>0.42</td>
<td>17.92</td>
<td>57.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td>0.7</td>
<td>0.77</td>
<td>0.42</td>
<td>17.92</td>
<td>57.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td>0.7</td>
<td>0.77</td>
<td>0.42</td>
<td>17.92</td>
<td>57.18</td>
</tr>
<tr>
<td>16:55</td>
<td>19:04</td>
<td>wide passage</td>
<td>0.1</td>
<td>0.9</td>
<td>0.67</td>
<td>0.20</td>
<td>19.42</td>
<td>41.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>1.0</td>
<td>0.67</td>
<td>0.20</td>
<td>19.42</td>
<td>41.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td>1.0</td>
<td>0.67</td>
<td>0.20</td>
<td>19.42</td>
<td>41.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td>1.0</td>
<td>0.67</td>
<td>0.20</td>
<td>19.42</td>
<td>41.89</td>
</tr>
<tr>
<td>16:55</td>
<td>19:04</td>
<td>S-shaped patrol</td>
<td>0.1</td>
<td>0.8</td>
<td>0.76</td>
<td>0.18</td>
<td>25.22</td>
<td>52.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>1.0</td>
<td>0.76</td>
<td>0.18</td>
<td>25.22</td>
<td>52.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td>0.9</td>
<td>0.76</td>
<td>0.18</td>
<td>25.22</td>
<td>52.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td>1.0</td>
<td>0.76</td>
<td>0.18</td>
<td>25.22</td>
<td>52.02</td>
</tr>
</tbody>
</table>
The second and third type of synthesized noise are related to the odometry source which is used by AMCL as incremental positioning. As mentioned above, in the baseline experiments without synthesized noise, the odometry was derived from the incremental positioning on the known trajectory. In reality, we do not have this ground truth information and therefore we wanted to simulate realistic odometry noises on top of the above mentioned incremental positioning. Two sources of noise account for odometry errors: drift and slippage [43]. We decided to model the odometry noises as two independent gaussian random processes in both x and y axes distributed $\mathcal{N}(\mu_x, \sigma_x^2)$ and $\mathcal{N}(\mu_y, \sigma_y^2)$ respectively such that the standard deviations $\sigma_x, \sigma_y$ reflect the slippage and the mean values $\mu_x, \mu_y$ reflect the drift. For symmetry reasons, we decided to examine the addition of odometry noise in x axis only. We first examined the slippage effect in a set of experiments in which we modified $\sigma_x$ in the range $[0.01, 0.05]$. The results are in Table 20 (with $\sigma_x = 0$ as reference). Then, in order to examine the effect of odometry drifts in addition to slippage, we set $\sigma_x = 0.01$ and modified $\mu_x$ in the range $[0.001, 0.005]$. The results, as well as the reference experiments of $\sigma_x = 0.01, \mu_x = 0$, are in Table 21. In both sets of experiments, like in the case of measurement noise, the canopies instances demonstrated stability whereas the trunks instances demonstrated gradual degradation in all measures as noise parameter increases.
<table>
<thead>
<tr>
<th>map image</th>
<th>tracking image</th>
<th>trajectory</th>
<th>sigma_x</th>
<th>convergence rate</th>
<th>in-band rate</th>
<th>settling time</th>
<th>uncertainty</th>
<th>variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:08</td>
<td>15:53</td>
<td>wide passage</td>
<td>0.01</td>
<td>1.0</td>
<td>0.9</td>
<td>0.53</td>
<td>0.29</td>
<td>28.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02</td>
<td>1.0</td>
<td>0.9</td>
<td>0.54</td>
<td>0.27</td>
<td>27.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.03</td>
<td>1.0</td>
<td>0.9</td>
<td>0.50</td>
<td>0.12</td>
<td>29.49</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.04</td>
<td>1.0</td>
<td>0.9</td>
<td>0.41</td>
<td>0.08</td>
<td>32.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
<td>0.8</td>
<td>0.11</td>
<td>0.32</td>
<td>0.01</td>
<td>35.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S-shaped patrol</td>
<td>0.01</td>
<td>1.0</td>
<td>0.9</td>
<td>0.89</td>
<td>0.76</td>
<td>17.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02</td>
<td>1.0</td>
<td>0.9</td>
<td>0.88</td>
<td>0.73</td>
<td>20.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.03</td>
<td>1.0</td>
<td>0.9</td>
<td>0.84</td>
<td>0.79</td>
<td>26.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.04</td>
<td>0.9</td>
<td>0.8</td>
<td>0.82</td>
<td>0.47</td>
<td>5.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
<td>0.9</td>
<td>0.9</td>
<td>0.68</td>
<td>0.35</td>
<td>48.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fork-shaped patrol</td>
<td>0.01</td>
<td>1.0</td>
<td>0.7</td>
<td>0.66</td>
<td>0.29</td>
<td>23.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02</td>
<td>1.0</td>
<td>1.0</td>
<td>0.54</td>
<td>0.28</td>
<td>31.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.03</td>
<td>1.0</td>
<td>1.0</td>
<td>0.65</td>
<td>0.33</td>
<td>23.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.04</td>
<td>1.0</td>
<td>0.7</td>
<td>0.64</td>
<td>0.18</td>
<td>24.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
<td>0.9</td>
<td>0.2</td>
<td>0.18</td>
<td>0.00</td>
<td>54.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>narrow passage</td>
<td>0.01</td>
<td>1.0</td>
<td>0.8</td>
<td>0.71</td>
<td>0.05</td>
<td>53.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02</td>
<td>1.0</td>
<td>0.9</td>
<td>0.77</td>
<td>0.42</td>
<td>17.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.03</td>
<td>0.9</td>
<td>0.9</td>
<td>0.77</td>
<td>0.49</td>
<td>18.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.04</td>
<td>0.8</td>
<td>0.8</td>
<td>0.76</td>
<td>0.44</td>
<td>19.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
<td>0.8</td>
<td>0.8</td>
<td>0.73</td>
<td>0.31</td>
<td>22.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.06</td>
<td>0.9</td>
<td>0.9</td>
<td>0.72</td>
<td>0.22</td>
<td>23.96</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.07</td>
<td>1.0</td>
<td>0.9</td>
<td>0.70</td>
<td>0.12</td>
<td>27.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.08</td>
<td>1.0</td>
<td>1.0</td>
<td>0.72</td>
<td>0.60</td>
<td>16.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.09</td>
<td>1.0</td>
<td>1.0</td>
<td>0.70</td>
<td>0.51</td>
<td>17.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
<td>0.9</td>
<td>0.9</td>
<td>0.71</td>
<td>0.49</td>
<td>17.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.11</td>
<td>0.9</td>
<td>0.9</td>
<td>0.68</td>
<td>0.35</td>
<td>18.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.12</td>
<td>0.9</td>
<td>0.8</td>
<td>0.38</td>
<td>0.09</td>
<td>35.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.13</td>
<td>0.9</td>
<td>0.8</td>
<td>0.56</td>
<td>0.07</td>
<td>25.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>narrow passage</td>
<td>0.01</td>
<td>1.0</td>
<td>0.9</td>
<td>0.67</td>
<td>0.20</td>
<td>19.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02</td>
<td>1.0</td>
<td>1.0</td>
<td>0.65</td>
<td>0.23</td>
<td>20.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.03</td>
<td>1.0</td>
<td>0.3</td>
<td>0.62</td>
<td>0.05</td>
<td>22.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.04</td>
<td>0.8</td>
<td>0.0</td>
<td>0.27</td>
<td>0.00</td>
<td>38.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
<td>1.0</td>
<td>0.0</td>
<td>0.54</td>
<td>0.00</td>
<td>28.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>wide passage</td>
<td>0.01</td>
<td>1.0</td>
<td>1.0</td>
<td>0.8</td>
<td>0.85</td>
<td>14.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02</td>
<td>1.0</td>
<td>0.7</td>
<td>0.83</td>
<td>0.36</td>
<td>15.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.03</td>
<td>1.0</td>
<td>0.8</td>
<td>0.80</td>
<td>0.31</td>
<td>16.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.04</td>
<td>1.0</td>
<td>0.7</td>
<td>0.70</td>
<td>0.04</td>
<td>27.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
<td>1.0</td>
<td>1.0</td>
<td>0.73</td>
<td>0.05</td>
<td>21.71</td>
</tr>
</tbody>
</table>

Table 20: aggregated AMCL results (slippage noise)
<table>
<thead>
<tr>
<th>map image</th>
<th>tracking image</th>
<th>trajectory</th>
<th>$\mu_x$</th>
<th>convergence rate</th>
<th>in-band rate</th>
<th>settling time</th>
<th>uncertainty</th>
<th>variance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>canopies trunks</td>
<td>canopies trunks</td>
<td>canopies trunks</td>
<td>canopies trunks</td>
<td>canopies trunks</td>
</tr>
<tr>
<td>15:08</td>
<td>15:53</td>
<td>wide passage</td>
<td>0</td>
<td>1.0 0.003 0.002</td>
<td>28.00 127.61</td>
<td>40.78 260.95</td>
<td>1.39 31.44</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001</td>
<td>1.0 0.003 0.002</td>
<td>28.22 137.61</td>
<td>40.78 260.95</td>
<td>1.39 31.44</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.002</td>
<td>1.0 0.003 0.002</td>
<td>29.82 136.98</td>
<td>44.73 321.40</td>
<td>1.29 32.42</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.003</td>
<td>1.0 0.003 0.002</td>
<td>30.25 137.39</td>
<td>51.95 315.96</td>
<td>1.62 7.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.004</td>
<td>1.0 0.003 0.002</td>
<td>32.19 144.10</td>
<td>8.11 365.75</td>
<td>1.78 16.89</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.005</td>
<td>0.9 0.003 0.002</td>
<td>35.20 181.78</td>
<td>5.69 531.28</td>
<td>6.79 14.90</td>
<td></td>
</tr>
<tr>
<td>15:08</td>
<td>15:53</td>
<td>S-shaped patrol</td>
<td>0</td>
<td>1.0 0.003 0.002</td>
<td>20.29 36.97</td>
<td>36.97 82.91</td>
<td>0.22 8.35</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001</td>
<td>1.0 0.003 0.002</td>
<td>22.99 39.20</td>
<td>31.46 78.07</td>
<td>0.38 7.22</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.002</td>
<td>1.0 0.003 0.002</td>
<td>26.19 42.65</td>
<td>44.04 116.42</td>
<td>0.58 9.21</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.003</td>
<td>1.0 0.003 0.002</td>
<td>32.19 144.10</td>
<td>8.11 365.75</td>
<td>1.78 16.89</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.004</td>
<td>1.0 0.003 0.002</td>
<td>35.20 181.78</td>
<td>5.69 531.28</td>
<td>6.79 14.90</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.005</td>
<td>0.9 0.003 0.002</td>
<td>38.00 191.50</td>
<td>51.95 315.96</td>
<td>1.62 7.65</td>
<td></td>
</tr>
<tr>
<td>15:53</td>
<td>16:55</td>
<td>wide passage</td>
<td>0</td>
<td>1.0 0.003 0.002</td>
<td>23.83 157.33</td>
<td>51.58 360.01</td>
<td>2.94 33.29</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001</td>
<td>0.8 0.003 0.002</td>
<td>40.62 235.50</td>
<td>53.23 456.70</td>
<td>8.09 14.90</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.002</td>
<td>0.8 0.003 0.002</td>
<td>23.58 144.10</td>
<td>54.42 342.46</td>
<td>3.54 33.42</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.003</td>
<td>0.8 0.003 0.002</td>
<td>24.57 165.21</td>
<td>48.38 143.46</td>
<td>3.17 36.94</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.004</td>
<td>0.8 0.003 0.002</td>
<td>24.88 163.19</td>
<td>49.48 430.48</td>
<td>4.30 33.90</td>
<td></td>
</tr>
<tr>
<td>15:53</td>
<td>16:55</td>
<td>fork-shaped patrol</td>
<td>0</td>
<td>1.0 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.002</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.003</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.004</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td>16:55</td>
<td>19:04</td>
<td>narrow passage</td>
<td>0</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.002</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.003</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.004</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td>16:55</td>
<td>19:04</td>
<td>wide passage</td>
<td>0</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.002</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.003</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.004</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td>16:55</td>
<td>19:04</td>
<td>S-shaped patrol</td>
<td>0</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.002</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.003</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.004</td>
<td>0.8 0.003 0.002</td>
<td>19.90 150.06</td>
<td>22.44 122.57</td>
<td>0.56 34.62</td>
<td></td>
</tr>
</tbody>
</table>
5.1.2 Canopy-Based ICP

Another possible use of the canopies virtual scan is as an alternative odometry source. As demonstrated in chapter 2.1.1, the UGV’s wheel odometry estimation is poor, especially in turning points. ICP estimation based on LiDAR readings is also unsatisfactory.

In this chapter we bring one proof-of-concept for ICP estimation based on canopies virtual scans. As opposed to the canopy-based AMCL presented above, ICP deals with incremental positioning and does not achieve global localization. For this demonstration, we chose one virtual fork-shaped patrol trajectory (Figure 52). The virtual canopies scan generated along this trajectory are used as inputs to the ICP algorithm which yielded the state estimation illustrated in Figure 53.

Qualitatively, the ICP estimation resembles the fork-shaped patrol pattern. Furthermore, ICP did not fail this time at the turning points. As already mentioned, LiDAR readings from a UGV suffer from the effect of terrain bumps and these eventually affect the state estimation. This proof-of-concept emphasizes the advantage of the virtual canopies scans which do not suffer from this effect.

![Figure 52: fork-shaped trajectory](image-url)
Figure 53: ICP estimation
5.2 **HIGH ALTITUDE**

In many cases, the continuous acquisition of top-view images is not feasible. In this chapter, we suggest a group of architectures which opportunistically leverage periodic top-view images acquired at high altitude and capture the UGV and the entire orchard plot. The processing of the top-view images in the architectures below rely on the computer vision procedures presented in chapter 4.

5.2.1 **UGV Periodic State Update**

As already stated, a typical approach for UGV localization is the EKF which fuses information from the various sensors mounted on the UGV and estimates the vehicle’s pose. However, as shown in chapter 2.1.1, EKF estimation drifts over time and in orchards, where navigation areas are large, the accumulated errors are significant. In this architecture, we suggest using top-view images opportunistically to update the UGV’s estimated pose. Using the image alignment procedure suggested in chapter 4.2.1 and by segmenting the UGV from the images, it is possible to determine the relative translation of the vehicle and update the EKF state estimator on the UGV accordingly.

To demonstrate the feasibility of the suggested concept, we use real data acquired from the Jackal UGV and from the Mavic Air UAV. In the November experiment, we teleoperated the Jackal in the orchard along the trajectory roughly sketched in Figure 54 and recorded its telemetry throughout the entire maneuver. In each of the eight waypoints marked in the figure, we took a snapshot from the Mavic Air camera. The Jackal started and ended its maneuver at the same point (marked in red) which is also the last waypoint among the eight mentioned above. We calculate the estimation error at the end point, where we know the ground truth pose is identical to the initial pose.

The Jackal uses ROS’s EKF, implemented in the robot localization package [44], as a state estimator. The Jackal’s EKF fuses information from its wheel odometry and its IMU. The ROS EKF module exposes a “set_pose” service which allows updating the absolute estimated pose to a given fixed value. We used this service to update the pose based on the top-view information. The segmentation of the vehicle from the top-view image was done manually and the scaling of pixels to meters is done according to the method described in appendix 9.3.

We conducted eight offline experiments on the collected data, with a single “set_pose” update in each, according to the eight waypoints mentioned above. These experiments examine the effect of an update on the final estimation error and are compared against the baseline EKF instance without any update (Table 22). The results demonstrate significant improvement in most of the experiments. In Figure 55 we plot the trajectories of estimated poses in all experiments: the baseline EKF instance in black and the updated EKF instance in pink (the dashed part illustrates the update). The red point denotes the starting point and the black and pink points denote the end points of the two EKF instances.
Figure 54: rough sketch of the Jackal’s trajectory

Table 22: final EKF estimation errors

<table>
<thead>
<tr>
<th>update in</th>
<th>baseline EKF error [m]</th>
<th>updated EKF error [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>14.33913277</td>
<td>5.020617198</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>4.63378149</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>14.15793896</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>14.30411048</td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>14.30411048</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>14.30411048</td>
</tr>
<tr>
<td>G</td>
<td></td>
<td>14.30411048</td>
</tr>
<tr>
<td>H</td>
<td></td>
<td>3.175799583</td>
</tr>
</tbody>
</table>
Figure 55: baseline EKF vs updates EKF (titles denote the update point)
5.2.2 Semantic Global Path Planning
The semantic understanding of tree positions from the top-view observation of the orchard allows to plan semantic paths of the UGV. In other words, the trunk positions approximated by Algorithm 3 can be used as semantic waypoints for global path planning of the UGV. This capability is, as already mentioned, crucial for precision agriculture tasks. Obviously, a local path planner and a localization mechanism are complementary to the global path planner in order to execute any navigation task. In this chapter we bring a proof-of-concept for semantic global path planning using a cost map of the orchard.

5.2.2.1 Orchard Cost Map
The major consideration for UGV movement in an orchard is its distance from the trees. It is generally desired that the UGV stays away from trees, with which canopies it might interfere. In addition, the terrain is usually easier at the center of inter-row passages and that is an additional motivation for driving the vehicle far from the trees. However, crossing a row in between the trees can significantly shorten driving time in certain conditions, and providing that driving is done at a low velocity, it can be safe for both the vehicle and the trees.

We chose to apply the canopy contours extraction procedure (Algorithm 1) on high altitude top-view images to create the orchard cost map. The map is designed such that a cell’s cost increases the deeper it is in the contours’ interior. The map values range in [0, 1] where 0 denotes complete passability and 1 denotes a non-passable cell. We chose to build the cost map as gradual stripes of costs, relative to the canopy contours (Figure 56). This map representation encourages driving in the middle of the passages and prevents driving near the canopy centroids on the one hand. On the other hand, this map allows a certain degree of freedom to cross in between trees.
5.2.2.2 Global Planning

As a path planner, we chose to use the common A-star algorithm [45]. In our A-star implementation, a node is a pixel of the cost map and its adjacent nodes are the pixel’s eight neighboring pixels. Valid neighbors are ones whose pixel value is lower than 1. The distance between adjacent nodes is comprised of a constant 1 and an additional cost which stems from the cost map. Algorithm 23 describes the function which generates adjacent nodes and calculates their distances. The heuristic function we use for A-star is a simple Euclidean norm. This heuristic function is not always optimistic, for instance between two diagonal zero valued pixels, and it can consequently lead to suboptimal solutions. Nevertheless, we empirically found that the resulting paths are reasonable and smoother using this function.

Algorithm 23: Get-Adjacent-Nodes-Distances

<table>
<thead>
<tr>
<th>Input:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• cost_map [numeric matrix]</td>
<td></td>
</tr>
<tr>
<td>• node [pixel coordinates]</td>
<td></td>
</tr>
<tr>
<td>Output: adjacent_nodes_distances [dictionary of pixel coordinates → float]</td>
<td></td>
</tr>
</tbody>
</table>

adjacent_nodes_distances := {}
neighboring_pixels := Get-Eight-Neighboring-Pixels(node)
for neighbor in neighboring_pixels:
    if cost_map[neighbor] ≠ 1:
        distance := 1 + cost_map[neighbor]
        adjacent_nodes_distances[neighbor] := distance
In Figure 57 we demonstrate three semantic paths on the cost map. Each source and target point in this demonstration is determined by the center point between a pair of trunk positions extracted using Algorithm 3. In all three examples, the planner prefers the darker areas of the map, yet, in certain cases the path crosses gray areas in between the trees to shorten the driving time.
A: from (2/G, 3/G) to (3/C, 4/C)

B: from (3/D, 4/D) to (7/D, 8/D)

C: from (3/A, 4/A) to (8/D, 9/D)

Figure 57: semantic path planning
5.2.2.3 Path Update

To further demonstrate the advantage of global path planning using a top-view image, we consider the following hypothetic scenario: a UGV is assigned a series of tasks in specific locations in the orchard. The obvious first step before executing the tasks is finding short paths between the desired waypoints. One top-view image can help finding those paths using the global path planner suggested above. For this demonstration we consider ten arbitrary waypoints, labeled A to J. The waypoints along with the connecting trajectories, appear in Figure 58.

In practical scenarios, it is not guaranteed that a single UGV is the only operator in the orchard plot. Additional vehicles or static objects might block certain passages in the orchard and thus interfere with the UGV's movement. Periodic top-view images of the orchard plot can help identifying such obstacles and update the planned global path accordingly. In Figure 59 we placed an artificial obstacle in two different locations along the originally planned path. By aligning the new images against the one acquired earlier (Figure 58), it is possible to detect significant changes between the images which are likely to indicate obstacles. The alignment can be done using the approximated trunks as described in chapter 4.2.1. In this demonstration the artificial obstacles were manually marked, however, today's computer vision techniques allow to detect the changes automatically. Following the obstacle detection, new paths are calculated. Figure 59-A demonstrates an image captured once the UGV hypothetically reaches waypoint C which results in path update between waypoints D and E. Figure 59-B demonstrates an image captured once the UGV hypothetically reaches waypoint F which results in path update between waypoints I and J. These two examples demonstrate how periodic top-view image can help update the planned path and by that shorten the UGV's driving time.

Figure 58: series of waypoints and their connecting paths
A: obstacle in third passage to the right

B: obstacle in fifth passage to the right

Figure 59: path updates to avoid obstacles
In this work we demonstrate how the addition of top-view images can aid in localization, mapping and path planning tasks of a ground mobile robot in orchards. By means of computer vision, we are able to provide additional information from the air to the ground vehicle. Besides the top-view images, some of our suggested methods assume additional perceptions means on the ground robot, but they are all independent of GPS, which is known to be unreliable and inaccurate in orchards. In addition, we refrain from using artificial landmarks in the field.

Two main computer vision procedures were developed in this work and were used throughout it. The canopies extraction procedure constructs a binary mask of canopies. This is practically a simplified representation that is used for localization, mapping and path planning purposes. The trunks approximation procedure, which relies on canopies extraction, provides even a more compact representation which we use for several navigation architectures. We proved in this work that these two procedures are insensitive to variations in the top-view image throughout the day and are thus considered reliable and invariant. At the second part of the work, we demonstrated various navigation architectures which leverage these two procedures.

We presented the idea of virtual canopies scans and their use for global localization (canopy-based AMCL) and incremental positioning (canopy-based ICP). We profoundly examined how the virtual canopies scans help addressing the kidnapped robot problem in orchards using AMCL. The successful AMCL results in the numerous synthesized experiments utterly verify our assumption that canopy shapes constitute unique tree signatures. This conclusion implies a great advantage for the canopies virtual scans over the typical LiDAR readings at ground level.

We also demonstrated how periodic top-view images can serve for periodic state update of a vehicle navigating in the orchard. In addition, we presented the idea of an orchard cost map that is extracted from the top-view image. We demonstrated how this map can serve for global path planning and for path update.

We see two main contributions of this study to precision agriculture in orchards. First, the trunks approximation procedure allows us to identify individual elements, i.e. trees, in the orchards and label them. With the ability we demonstrated to plan a path to specific trees, it is possible provide individual treatment (e.g., irrigation or spraying) to individual trees. Second, the centimeter-level localization accuracy that we demonstrated in the canopies-based AMCL chapter makes it possible to reach the desired tree in the orchard accurately.

6.1 Future Work
The computer vision methods in this work were developed for almond orchards. Other types of trees might have different visual characteristics such as canopy shapes and trees spacing, and therefore our techniques might be unsuitable for other types of orchards. Deep Learning is commonly used today for computer vision tasks such as segmentation and object detection and might be helpful for canopies extraction and trunks approximation as well. The big advantage of deep learning is its generalization
ability, the disadvantage though is the need of a large annotated dataset in order to train a model whereas our method does not require any prior annotation.

Another research direction that is related to our computer vision procedures is temporal map update. For this work, we assumed that one image is used to generate a map which does not change from that point onwards. It is possible to suggest a map update procedure which takes into account multiple images acquired at different times and averages them, possibly with decaying weights per time of acquisition. This map is likely to better represent the orchard.

The successful results of canopy-based AMCL inspires the use of AMCL for UAV localization as well. The formulation of this problem is slightly different as it requires estimating not only the UAV’s horizontal pose \((x,y)\) but also its heading \(\theta\) and its altitude \(z\). However, we believe that the canopy signatures can help addressing this problem as well. Another idea inspired by AMCL is path planning that is intended to minimize the uncertainty. We believe that machine learning models can observe AMCL’s behavior and learn how to generate trajectories which avoid high uncertainty.

An additional future direction is related to UAV path planning. When one UAV serves in multiple orchard plots in turns, its trajectory should be carefully planned to achieve maximal coverage while considering its flight time limitations.

In this work we assumed one UGV in each plot. Multiple ground agents can pose opportunities but also challenges. We believe that aiding top-view can be helpful to achieve better coverage and utilization of the ground robots. Another assumption that we have made is planar orchards. In non-planar orchard plots, mapping and state estimation are more complex and this could be an additional research direction.
7 Bibliography


9 APPENDICES

9.1 THE NEGLIGIBLE EFFECT OF THE KEystone DISTORTION
For our trunks approximation procedure in chapter 04.2, we assume that the trees are placed on a transformed grid. We assume a linear affine transformation and neglect the keystone distortion. In Figure 60, we manually marked parallel lines and fit them to the tree rows. The Keystone distortion explains parallel lines in the real world that do not appear parallel in the image. The red lines in the figure fit the tree rows rather well and assuming that the rows are nearly parallel in the real world, we can say that the Keystone distortion is not significant in our case.

Figure 60: parallel lines manually fitted to the tree rows
9.2 Trunks Approximation and Semantic Labeling — Additional Results

The figures in chapter 4.2 visualize the trunks approximation procedure on an image acquired in the April experiment at 15:08 and using the layout in Figure 20. Below are additional examples of trunks approximation. The first five are of images of the same plot described by Figure 20, acquired in April and November. The two examples that follow are images taken in another plot during the November experiment. The layout of that plot is defined in Figure 61:

Figure 61: plot layout

Figure 62: April, 15:53
Figure 65: November, 10:09

Figure 66: November, 10:10
9.3 Pixels to Meters Scaling

In order to use the top-view image information for navigation tasks, any computer vision output in pixels must be converted to meters at the ground level. As the top-view images are acquired in varying altitudes, a generic method is required for inferring the pixel-to-meter ratio in every image.

As a preliminary stage, we calculated the intrinsic parameters of the UAV’s camera by using a known checkerboard target [46]. We discovered that the focal lengths $f_x$ and $f_y$ are almost identical and therefore decided to neglect the focal distortion.

We chose to calculate the pixel-to-meter ratio as a quotient of known distances at the ground level and their matching distances in the image. In our experiments, we performed a few measurements of intra-row distances between trunks and inter-row distances as well. Then, we calculated the average values. In reality, orchard planting is done based on nominal inter-row and intra-row distances and the farmer can provide these numbers. The distances in pixels can be easily derived from the trunks approximation procedure elaborated in chapter 04.2; we chose to use the optimized $\Delta x$ and $\Delta y$ outputs of the Nelder-Mead optimization stage. We get two ratios:

\[ r_x = \frac{\Delta x}{\text{intra-row distances}} \]
\[ r_y = \frac{\Delta y}{\text{inter-row distances}} \]

Bearing in mind that $f_x \approx f_y$ in our case, we chose to calculate the pixel-to-meter ratio as the average of the two ratios calculated above:

\[ r = \frac{r_x + r_y}{2} \]
### 9.4 Parameter Values

Below are the parameter values used in the algorithms implemented in this work and in algorithms implemented by third-party libraries. For the third-party libraries, only non-default values are listed.

<table>
<thead>
<tr>
<th>Algorithm 1: Extract-Canopy-Contours</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPER_HSV_BOUND</td>
<td>(210, 100, 100)</td>
</tr>
<tr>
<td>LOWER_HSV_BOUND</td>
<td>(50, 5, 0)</td>
</tr>
<tr>
<td>CONTOUR_AREA_THRESHOLD</td>
<td>8000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 3: Approximate-And-Label-Trunks</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CROP_RATIO</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 4: Estimate-Tree-Rows-Orientation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGLE_INCREMENT</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 6: Estimate-Trees-Grid-Cell-Dimensions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q_PERCENTILE_X</td>
<td>50</td>
</tr>
<tr>
<td>Q_PERCENTILE_Y</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 7: Estimate-Trees-Grid-Shear</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q_PERCENTILE</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 11: Optimize-Trunks-Grid</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GRID_DIMS_MARGIN</td>
<td>60</td>
</tr>
<tr>
<td>ORIENTATION_MARGIN</td>
<td>7</td>
</tr>
<tr>
<td>SHEAR_MARGIN</td>
<td>0.12</td>
</tr>
<tr>
<td>TRANSLATION_MARGIN</td>
<td>60</td>
</tr>
<tr>
<td>SIGMA_MARGIN</td>
<td>50</td>
</tr>
<tr>
<td>INITIAL_SIGMA_TO_DIM_Y_RATIO</td>
<td>0.33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 13: Refine-Trunk-Positions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WINDOW_EXPANSION</td>
<td>1.1</td>
</tr>
<tr>
<td>SIGMA_SCALE</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 15: Generate-Virtual-Canopies-Scan</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER_OF_SAMPLES</td>
<td>360</td>
</tr>
<tr>
<td>MIN_RADIUS</td>
<td>3</td>
</tr>
<tr>
<td>MAX_RADIUS</td>
<td>250</td>
</tr>
<tr>
<td>SEARCH_DELTA</td>
<td>2</td>
</tr>
</tbody>
</table>

**AMCL**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ODOM_ALPHA1</td>
<td>0.05</td>
</tr>
<tr>
<td>ODOM_ALPHA2</td>
<td>0.05</td>
</tr>
<tr>
<td>ODOM_ALPHA3</td>
<td>0.1</td>
</tr>
<tr>
<td>ODOM_ALPHA4</td>
<td>0.05</td>
</tr>
<tr>
<td>ODOM_ALPHA5</td>
<td>0.03</td>
</tr>
<tr>
<td>MIN_PARTICLES</td>
<td>1000</td>
</tr>
<tr>
<td>MAX_PARTICLES</td>
<td>20000</td>
</tr>
</tbody>
</table>
רובוטים ניידים הולכים ונעשים נפוצים בחקלאות והם משמשים למגוון מטרות. במטעים ישנן משימות מגוונות שרובוט נייד יכול לבצע כגון ח℠shaft של מזיקים, הערכה יבול או ריסוס סלקטיבי. המשותף לכל המשימות הללו הוא הדרוש ניווט אוטונומי ומדויק. עבור משימות מסוימות, הصرف הגסה של ממוקם הוא מספק. לעומת זאת, במשימות אחרות, נחוצה רמת דיוק של סנטימטרים בודדים על מנת לאפשר יכולת ח℠shaft או יכולת מניפולציה מדויקות, קרי ח℠shaft מדם. לדוגמה, יכולת ניווט מדויקת של רכב קרקעי בלתי מאויש מאפשרת לבצע ריסוס סלקטיבי ו.TXT מטעם ضد מזיקים על בסיס מצבו של העץ והמידע ההיסטורי עליו. מטרת העל של עבודה זו היא להציע ת…”

 Fresno 2023-02-12 16:00 03

 – Fresno 2023-02-12 16:00 03

 Fresno 2023-02-12 16:00 03
המחקר נעשה בהנחיית פרופ' אמירDegani, באגף התכنية הבינה-
יחידה למשרוכות אוטונומיות וробוטיקה.

רשימת פרסומים

חלקים מתוך חיבור זה הוצגו בכנסים בתקופת המחקר של המחבר:


חלקים מתוך חיבור זה מעובדים למאמר של המחבר:

O. Shalev and A. Degani, “Canopy-Based Localization in Orchards Using Top-View Imagery” (in preparation).
גיוט רובוטים במשטחים בסיעת מבט - על

חיבור על מחקר
לש"מ מילוי חלף של הדירישות לрабבלת התיאור מגיסטר למדעי במערכות
אוטונומיות ורובוטיקה

עומר שלֶל

הוגש לסנט הטכניון - מרכז טכנולוגי לישראל
新西ו החמשע"ט חיפה יוני 2019
גיוס רוביוטים במטעיים לסירוג ממבת-על

עומר של"א