Autonomous Drone Mapping in Orchard Environments

Ke Wang Moein Taherkhani Elias Marks Federico Magistri

***Abstract*— While a continually growing world population necessitates a steady increase in food production, agriculture is facing a decline in human work-force. This trend motivates the deployment of autonomous mobile robots to undertake labor-intensive tasks. Effectivity of this solution however, depends on the understanding of robots from the geometry and semantics of agricultural environments, raising the need for maps. However, the cluttered nature of these environments introduces caveats to a data-collection/mapping platform: (i) difficulties in self-localization, (ii) extracting semantic information from the scene, and (iii) fusing information to build a final map. In this work, we assemble available popular pipelines that tackle each of these problems separately, forming an end to end mapping pipeline. Our building blocks include a visual SLAM pipeline and an instance segmentation pipeline, followed by a final panoptic mapping pipeline. We will explore and test these separate blocks, trying to identify limiting factors to the entire pipeline and providing a discussion about the final output.**

I. Introduction

As a result of a steady growth in world population, there is an ever-growing need to produce more agricultural products. Current agriculture sector, meanwhile facing a decline in human work-force, needs a boost in productivity in order to cope with this need. Therefore, use of autonomous robots for performing labor-intensive and time-consuming tasks gains growing appeal as a resolution. The successful operation of robots in agricultural fields is tied to how well they can perceive and interact with the environment. In orchard environments, a map is necessary for tasks like automated monitoring, robotic harvesting, and pruning. This map needs to provide robots with information about the 3D structure as well as semantics of the scene. In order to obtain a suitable map, a data-collection platform (e.g. a UAV or UGV) typically moves in between rows of trees to get a full view of the canopies, branches, and fruits. The cluttered structure of these environments however, poses challenges to a one such mapping pipeline. Our approach first tailors two sets of datasets, namely, real-world and synthetic, for our experiments. We proceed by breaking down the mapping process into three major blocks, i.e., localization of the platform, extracting semantic and instance information from the scene, and building of a final panoptic map. Meanwhile, we explore the challenges associated with each block in our pipeline. Lastly, we assemble three existing examples, one from each block, to form a panoptic mapping pipeline.

For reasons of convenience in data gathering, as well as free and unlimited deployments of the setup, we use a simulation environment first. We create a mesh-based 3D apple orchard in Microsoft's Airsim [16]. This environment is populated by replicas of the same tree and apple objects and provides us with photorealistic 2D images along with depth maps. This simulation also provides us with ground truth information on poses as well as semantic masks.

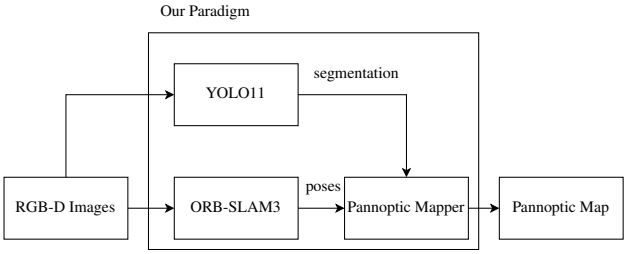


Fig. 1: Illustration of our paradigm

Our real-world data are RGB-D images recorded in between rows of apple trees in an orchard. In the first sequence camera points tangent to the trajectory (front view). The second sequence is from another trajectory, with the camera pointing perpendicular to the path and towards the apple trees. Ground-truth poses for both trajectories are obtained via LiDAR odometry and are available for evaluation.

First, we tackle the robot localization problem. Robot poses are a requirement for a panoptic map and their certainty is crucial to the quality of the map. To obtain poses, we leverage ORB-SLAM3 [1], a popular visual SLAM pipeline. We first experiment with this pipeline in simulation and discuss its underperformance, motivating us to shift the remainder of the experiments to real-world data. ORB-SLAM's performance on these data proves satisfactory and lays the ground for the remainder of experiments.

Secondly, we train the You-Only-Look-Once (YOLO) v11 [8] convolutional neural network on MinnieApple dataset's [7] annotated apple images. These data include segmentation masks for apple instances in different orchard scenes with lighting and scale variations. We then test this model, evaluate its performance and use it to draw inference on our RGB data. The final output of this stage is a set of instance segmentation masks for apples present in our RGB scenes. Throughout this step, we adhere to the COCO dataset [9] format and prepare the training, testing, and our inference data accordingly.

Finally, we leverage panoptic multi-TSDFs [15], using poses and instance segmentation masks from the previous steps. We delegate the instance tracking task to the mappers built-in projective tracking method which uses camera poses to track instances in between frames. Due to a lack of a reference map, the resultant panoptic map is then qualitatively discussed and speculated upon.

In short, the end goal of this work is to assemble an end-to-end automated pipeline which outputs a panoptic map of an apple orchard while using only a sequence of RGB-D data as input. Our general paradigm is illustrated in Fig. 1.

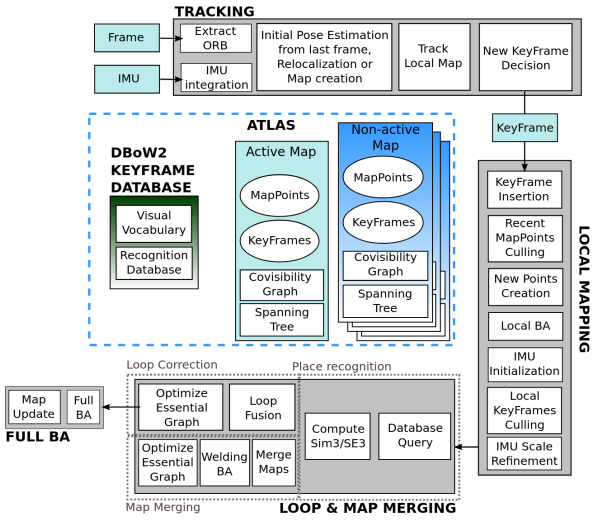


Fig. 2: Overview of ORB-SLAM3

II. Related Work

A. Visual SLAM

The approach LSD-SLAM by Engelet al. [2] proposes a direct feature-less monocular SLAM algorithm, which allows building large-scale, consistent maps of the environment. Along with accurate pose estimation based on direct image alignment, the 3D environment is reconstructed in real-time as pose-graph of keyframes with associated semi-dense depthmaps. The approach by Engelet al. [11] proposes ORB-SLAM, using ORB features whose descriptors provide short-term and mid-term data associations. ORB-SLAM builds a covisibility graph to limit the complexity of tracking and mapping, and performs loop closing and relocalization using the bag-of-words library DBoW2 [4], achieving long-term data association.

B. Segmentation

The approach by Long et al. [10] aims inference of image by convolutional network trained end-to-end, pixels-to-pixels. The approach (YOLO) by Redmon et al. [13] reframes object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities, processing images in real-time. The approach YOLO 10 by Wang et al. [17] aim to further advance the performance-efficiency boundary of YOLOs in both, the post-processing and the model architecture. They first present the consistent dual assignments for non-maximum suppression free training of YOLOs, which follows competitive performance and low inference latency. Moreover, they introduce the holistic efficiency-accuracy driven model design strategy for YOLO.

C. Mapping

In particular, volumetric representations such as occupancy [6] or Truncated Signed Distance Fields (TSDF) [12] have found a lot of success. By dividing the map into a dense grid of voxels, they are able to explicitly represent free space and differentiate between known and unknown regions in the map, which is crucial for online planning. The approach by Schmidet al. [15] present a method for panoptic multi-TSDF integration and map management for temporally consistent mapping during online operation.

III. Our Approach

In our approach we divided our task into four subsections. Firstly we develop an orchard environment in simulator AirSim in order to create a realistic simulation of an orchard and generate the data for testing our pipeline. Subsequently, we implement the ORB-SLAM3 on the data captured in the simulator in order to evaluate the poses generated from ORB-SLAM3. Next, we implement the convolution neural network You Only Look Once 11(YOLOv11) in order to generate instance segmentation masks for our real-world images. Finally, we implement a flexible multi-resolution volumetric map representation panoptic multi-TSDFs .

A. Build an orchard simulation environment

Simulator plays a key role in the development and optimization of precision agriculture technologies. In simulation, a virtual environment is built and tuned, aiming to replicate different real-world environment under different conditions. Simulator enables repeatable experiments, reducing cost and time, and facilitating acquisition of reference data.

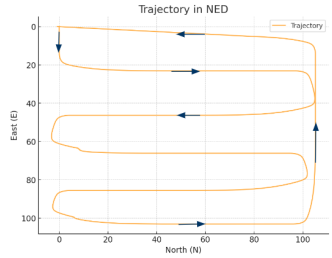
In our approach we employ one of the most popular simulator for autonomous vehicles AirSim as the simulator. AirSim is an open-source platform, includes a physics engine that can operate at a high frequency for real-time simulations and aims to narrow the gap between simulation and reality in order to aid development of autonomous vehicles. During simulation, the vehicle can be operated by keyboard or go through the waypoints which is provided as an input by launching simulator. In our approach, the waypoints are set between tree rows in order to obtain a full view of the orchard environment. Furthermore, the simulator provides the sensor data e.g. RGB image, depth image, gnss and imu etc. In our approach, we obtain the RGB images and depth image from the sensor as the input for the ORB SLAM3 and YOLOv11.

B. ORB-SLAM3

ORB-SLAM3 is a system built on ORB-SLAM2 and ORB-SLAM-VI and able to perform visual, visual-inertial and multi-map SLAM with monocular, stereo and RGB-D cameras, illustrated in Fig. 2. It consists of four components:

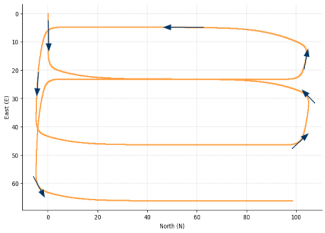
Atlas is a collection with an active map and non-activ maps. In the active map the tracking thread localizes the incoming frames, and is continuously optimized and grown with new keyframes by the local mapping thread. In addition, the system builds an unique DBoW2 database of keyframes that is used for relocalization, loop closing and map merging.

Tracking thread processes sensor information and computes the pose of the current frame with respect to the active map in real-time, minimizing the reprojection error of the matched map features ORB [14]. Moreover, this component addresses the problem of losing tracking. When the track is lost, the component attempts to relocalize the current frame

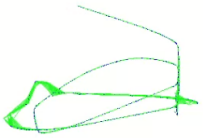


(a) reference trajectory with long loop

(b) estimated trajectory with long loop



(c) reference trajectory with short loop



(d) estimated trajectory with short loop

Fig. 3: Illustration of the AirSim reference trajectory and ORB-SLAM3 estimated trajectory with long loop(a, b) and with short loop(c, d)

into a map in Altas. If relocalized, tracking is resumed, switching the active map if needed. Otherwise, after a certain time, the active map is stored as non-active map in Atlas, and a new active map is initialized from scratch.

Local mapping thread adds keyframes and points into the active map, removes the redundant ones, and refines the map using visual or visual-inertial bundle adjustment, operating in a local window of keyframes close to the current frame.

Loop and map merging thread detects common regions between the active map and the whole Atlas when a keyframe is generated. If the common area belongs to the active map, it performs loop correction; if it belongs to a non active map, both maps are seamlessly merged into a single one, that becomes the active map. After a loop correction, a full bundle adjustment is launched in an independent thread to further refine the map without affecting real-time performance.

C. YOLOv11

YOLOv11 is the last iteration in the YOLO series of real-time object detection, instance segmentation and tracking. Building upon the advancements of previous YOLO versions, YOLOv11 introduces improvements in architecture and training methods, making it a versatile choice for computer vision tasks.

Similar to the other convolution neural network approach, we firstly need to generate the annotated images in order to train our YOLOv11 model which enable our model prediction and segmentation on the unknown images. However, annotating images require a significant amount of time. In our approach, we train the model through the use of benchmark dataset for apple detection and segmentation, MinneApple. In this dataset, over 41'0000 object instances are annotated in 1000 images with high resolution in orchard, where the lighting conditions and the ripeness of apples

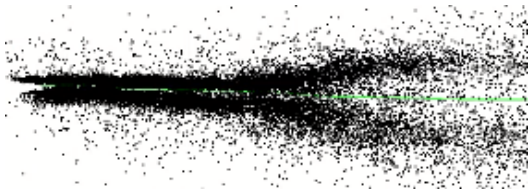


Fig. 4: ORB-SLAM3 scale drift in pure mono

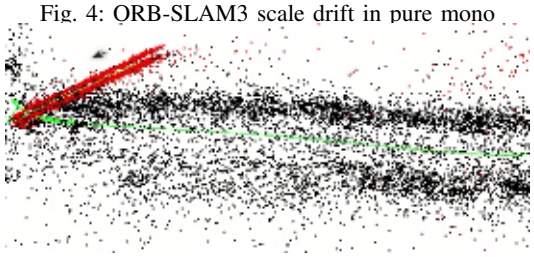


Fig. 5: ORB-SLAM3 scale reinitialization in pure mono

differ in different images, which enhance the model's ability to extract features from the images. Moreover, the dataset consist of the test dataset order to evaluate the performance of our model.

D. Pannoptic Mapping

In our approach we implement Panoptic Multi Truncated Signed Distance Function (TSDF) for mapping. In a nutshell and ideally, the outcome of this mapping pipeline should be a 3D voxel grid where every voxel stores information about occupancy (occupied/free), as well as class and instance of the occupying object. TSDF encodes the geometry of a scene by storing the signed distance to the nearest surface at each point in a voxel grid. A foundational understanding of TSDF is required to grasp Signed Distance Function (SDF), which returns the shortest distance to the closest surface. Meanwhile, the sign indicates whether the point is in the inside or outside of the object, positive for outside and vice versa. Moreover, TSDF truncates the distance values within a range $[-D, D]$ and the values are stored in a voxel grid, because the large distances are not relevant for surface reconstruction. Multiple observations from different viewpoints are combined in one value in order to integrate information to improve accuracy or to add missing patches of the surface, which is achieved by weighted summation and usually through iterative updates of the TSDF [18].

In order to capture the consistency and change of the instance in the real world, we divide the instance into objects, background and free space. For different object, the voxel size varies based on the complexity of the objects' geometry. The voxel size of free space is relatively larger to help store the map more efficiently. Every individual object, background or free space is associated in one submap. When the new incoming frame are tracked against the current map, all points in the active submap within the view frustum are projected into the image plane. Each input segment is associated to the submap that has the highest Intersection over Union (IoU) between predicted and rendered mask and the same class label.

IV. Experimental Evaluation

The main focus of this work is integration of a unified automatic pipeline which outputs a panoptic map of an apple orchard, while using only a sequence of RGB-D images as input. In doing so, we integrate three major building blocks that take care of the intermediate steps, i.e., localization of the platform, segmentation of apple instances, and building of a final panoptic map. Therefore, we provide experimental results and evaluations on our intermediate steps and discuss their caveats. Our final result however, is only qualitatively assessed and discussed upon.

The subsequent experiments align with our main expectations, namely: (i) ORB-SLAM3 tends to underperform in environments with highly repetitive pixel patterns. (ii) Scale drift and scale re-initialization in ORB-SLAM3 are present in case of pure monocular SLAM. Integrating depth information, should visibly improve this problem. (iii) possible domain gaps between training and inference visual data can impair the quality of a downstream panoptic map.

A. Experimental Setup

Simulation: For two major reasons, we were motivated to begin our experiments in a simulated environment. Firstly, simulators facilitate the convenient acquisition of noise-free ground-truth data, suitable for evaluation against a simulated platform. Secondly, incorporating arbitrarily complex trajectories in simulation proves feasible and cost-free compared to real-world domains, as it precludes wreckage of the platform and the need for manual handling or teleoperation. Accordingly, we chose Microsoft's AirSim [16] to simulate a mesh-based apple orchard environment and a UAV as our platform. We simulate our sole source of input data, the Intel RealSense D435 RGB-D camera, with configurable noise. Moreover, the simulator provides us with ground-truth data such as the poses of the platform, depth maps of the environment, and segmentation masks for objects of interest. Lastly, we note that we use identically rendered 3D tree and apple objects to populate the synthetic orchard, the effects of which are discussed in subsequent sections.

Real-world data: Our real-world data is comprised of two sequences of monocular RGB images and their corresponding depth maps, captured using an Intel RealSense d435 camera. One sequence features a front scene, where camera points tangent to the local trajectory and in the movement direction. In the other sequence, which is captured in a different orchard, the camera is directed perpendicular to the path and towards the row of apple trees. We will also use LiDAR odometry poses as reference for evaluation.

B. Performance

The first experiment evaluates the performance of our SLAM pipeline, ORB-SLAM3, in a simulated orchard. Results of this experiment support our first claim, i.e., ORB-SLAM3 is prone to poor trajectory estimation in highly repetitive environments. In this experiment, a UAV equipped with an Intel RealSense D435 RGB-D as front camera flies along trajectories defined by 3D waypoints. Two different

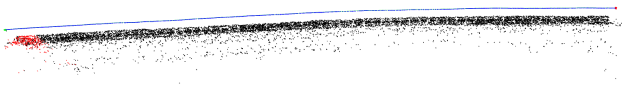


Fig. 6: scale consistency in ORB-SLAM3 trajectory and sparse map, including depth information

Fig. 7: Example of YOLO's poor instance segmentation on our data. As visible, many apple instances in this frame are missed.

waypoint trajectories are tested, incorporating both short and long-interval loop closures. The recorded RGB-D data are subsequently fed to ORB-SLAM3. For the purposes of this project, we discard the resultant sparse map of this SLAM pipeline and merely use the outcome poses for dense mapping in further sections. The resultant trajectories Fig. 3a 3b indicate that ORB-SLAM3 fails to detect loop closures in this environment, even when short-interval loop closures were incorporated (Fig. 3c 3d). The provision of concrete causality for these results is outside the scope of this work. However, we speculate that they stem from impaired data association, which is in turn caused by near-identical feature vectors extracted for actually non-identical feature points in the scene. This speculation is further bolstered in light of the presence of graphically identical objects in our simulation environment. Aforementioned results, along with the necessity of credible robot poses for building of a final map, motivate either a shift to real-world domain or fundamental changes to our simulated orchard. Due to time constraints, we choose the former.

The second experiment is carried out on a sequence of RGB images captured in a real-world apple orchard. In this experiment, we explore the feasibility of performing localization and trajectory estimation via solely using RGB images, excluding any depth information. We use the front-view RGB-D sequence for this experiment. The results of this experiment are aligned with our expectation that pure monocular ORB-SLAM is prone to scale drift within an active submap. Fig. 4 showcases this problem. The width of the pathway between the rows of trees is constant in reality. Whereas ORB-SLAM's solution suggests a gradually tapered row. This is in part explained by the fact that with monocular images, the reconstruction matrix can only be estimated up to a scale. Furthermore, this experiment showcases scale reinitialization in ORB-SLAM3 upon initialization of new maps and highlights that the offset between scales of consecutive submaps can be noticeable. This is visible in Fig. 5, where the black and the red parts of the map respectively denote the way forward (fixed map) and backward (still active submap) along the same path. This follows that the scales of trajectory and the sparse map should nearly be the same for both sections. However, a visible shift in scale is present, aligning with our expectations and motivating the incorporation of depth information for correct scale estimation.

Fig. 8: PanMap-TSDF's 3D geometric color map

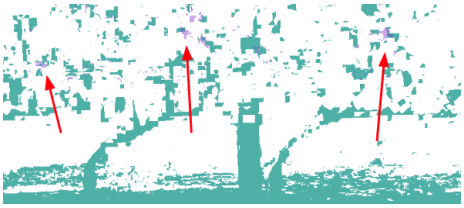
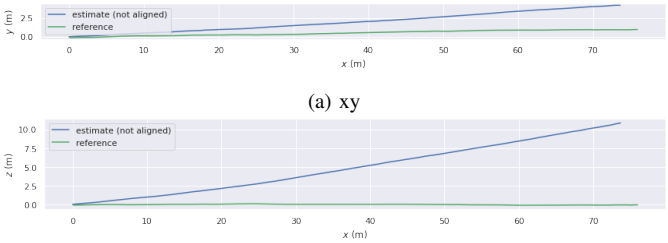


Fig. 9: Examples of poor instance tracking in PanMap-TSDF

We conclude exploring ORB-SLAM3 with a final experiment in which we incorporate both RGB and depth maps for the full SALM pipeline. This experiment is intended to provide us with reliable poses that we can further use for our mapping pipeline. Additionally, results align with our anticipation that incorporating depth information visibly alleviates the scale drift problem. We leverage depth maps in this experiment for estimating scale. The results from this experiments, whose metrics and comparisons with ground truth information are discussed in the next section, are consistent enough to be used as inputs for panoptic mapping further down in our goal pipeline. As before, these results include a sequence of camera poses along with a sparse point-cloud map. In this experiment however, we use side camera information instead of front camera, directed towards the tow of apple trees and perpendicular to the main trajectory. Fig. 6 indicates visible improvements in continual scale estimation and dimension consistency compared to the previous experiment (Fig. 4).

The next experiment pertains to our segmentation block. The objectives of this experiment are (i) to train the YOLOv11 model on the Minniapple dataset, ascertaining the output model performs reasonably with regards to available metrics, and (ii) to generate instance segmentation masks for apples on our RGB data. We follow a typical workflow for preparing the dataset, training, testing, and deploying a model. The Ultralytics API requires segmentation annotations as polygons, and as such we render the masks provided in MinneApple. Furthermore, we randomly bifurcate the annotated data into train and test sets, comprising respectively 80 and 20 percent of the total volume. We then train the YOLOv11M with 20.1 million parameters on the training set followed by inference on the test set to evaluate our model and draw certain metrics. As noted in the next section, these metrics demonstrate performance comparable to benchmark scores. Subsequently, we rely on this model and deploy it to our sequence of RGB images to segment instances of apples. It is of foremost importance to mention however,



(a) xy

(b) xz

Fig. 10: Ground truth trajectory vs ORB-SLAM3 estimate. (a) trajectory in the xy plane (parallel to ground), (b) trajectory in the xz plane, where z axis points upwards perpendicular to ground

that the obtained metric will not necessarily represent the performance on our data. In fact, we can't establish an assessment of segmentation results on our data, since these data lack annotations. A present domain gap in between training and our data is most likely to downgrade the final performance. An extreme example of poor segmentation on our data can be seen in Fig. 7.

We conduct the final experiment to obtain a panoptic map. Due to a lack of a reference panoptic map however, the results of this experiment are only sanity-checked. The input data to this stage are RGB images, depth maps, ORB-SLAM3's output poses, and YOLOv11's instance segmentation masks on apples. These data are arranged in ASL's FLAT dataset format. However, another very important input to this pipeline can be ground-truth tracking IDs of all instances. Alternatively, we simply provide unanimously labeled segmentation masks for all instances, delegating the tasks of individual instance tracking to PanMap's projective mechanism. We eventually launch Panmap-TSDF and the results are as follows:

(i) A 3D geometric map of the environment is shown Fig. 8. This figure is reasonable similar to the actual environment, suggesting that the voxel grid encodes grossly correct occupancy and color information.

(ii) A 3D panoptic map. However, this is not an ideal panoptic map. As shown in Fig. 9, multiple apple instances in disparate positions are incorrectly tracked as the same instance. These figures are only selected examples of this problem, who occurs almost throughout the map.

Even though the latter results prove rather inconclusive to the performance of PanMap, we can still speculate the most likely cause: poor performance of YOLOv11 as exemplified in Fig. 7. In addition to neglected instances, we observed inconsistent position of segmentation masks across consecutive frames which, can potentially mislead projective tracking.

As ground for future work, we can suggest the repetition of this experiment in a controlled setup with ground truth segmentation masks, confirming this speculated causality. If the results visibly improve, another step can be taken to provide ground-truth instance tracking IDs to measure the sole performance of PanMap's projective tracking mechanism.

C. Metrics

Finally, we provide common metrics for our SLAM and segmentation blocks.

TABLE I: Absolute position error statistics

Statistic	Value
Max	11.58
Min	0.19
RMSE	6.27

TABLE II: Segmentation metrics on MinneApple

mode	mAP[0.5:0.95]	mAP50
Mask RCNN [5]	0.433	0.763
Our trained YOLOv11	0.373	0.762

ORB-SLAM3: For evaluating the resultant trajectory of this pipeline, we compare it with the reference trajectory using the root mean square (RMS) of absolute translation error (ATE). The results in Tab. I belong to the experiments on real-world data where depth maps are incorporated for scale estimation, i.e. the third ORB-SLAM experiment. 2D plots in Fig. 10a and Fig. 10b indicate more drift in the z coordinate estimate, vertical to the plane of motion.

As for the first and second ORB-SLAM experiment however, conducted in simulation and in real-world without depth maps respectively, mere qualitative assessments indicated visible incompatibility with reference trajectories Fig. 3.

YOLOv11: Since we have non-annotated inference data from orchard, our segmentation metric are derived on the MinnieApple test set. Our mAP[0.5:0.95] [9] and mAP50 scores [3], shown in Tab. II, indicate close performance to that of Mask RCNN on MinneApple. Therefore, we rely on our trained model and proceed to draw inference on our data.

In summary, the metric for ORB-SLAM3 suggest outputs are well aligned with ground truth. This is further supported by the fact that the 3D geometric map chiefly resembles the actual environment. YOLO's metrics also indicate near-benchmark performance in the train/test domain, even though a visible but non-quantifiable downgrade exists upon shift to our inference data.

V. Conclusion

In this paper, we broached the problem of building a panoptic mapping in an apple orchard environment. We broke this problem down into parallel usage of a SLAM and an object segmentation pipeline, followed by a voxel grid mapping pipeline. We separately reviewed our building blocks and made assumptions about them, providing individual experiments and evaluations to validate these assumptions where feasible. Eventually, we inspected our final output map and discussed potential causes for non-ideal results. For potential future build up on this work, we suggest that:

(i) in case of simulating an environment for testing a visual SLAM pipeline, one should incorporate differently created 3D objects to avoid repetitive patterns, (ii) any provisional data, e.g., inertial, be incorporated for further improvement in the SLAM pipeline, (iii) while fixing all other factors, one tests the whole pipeline on a sequence of RGB data with ground-truth masks, (iv) one further explores PanMap-TSDF's projective object tracking mechanism, comparing its output against ground-truth object tracking IDs consistent with ground truth masks, (v) and a reference panoptic map be acquired for evaluating the final result.

References

[1] C. Campos, R. Elvira, J.J. Gomez, J.M.M. Montiel, and J.D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[2] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2014.

[3] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *Intl. Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.

[4] D. Galvez-Lopez and J.D. Tardos. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Trans. on Robotics (TRO)*, 28(5):1188–1197, 2012.

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn, 2018.

[6] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[7] N. Häni, P. Roy, and V. Isler. Minneapple: A benchmark dataset for apple detection and segmentation, 2019.

[8] G. Jocher, J. Qiu, and A. Chaurasia. Ultralytics YOLO, January 2023.

[9] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick. Microsoft COCO: Common Objects in Context. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2014.

[10] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[11] R. Mur-Artal, J. Montiel, and J.D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans. on Robotics (TRO)*, 31(5):1147–1163, 2015.

[12] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwar, and J. Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2011.

[15] L. Schmid, J. Delmerico, J. Schönberger, J. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena. Panoptic multi-tsdfs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8018–8024, 2022.

[16] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles, 2017.

[17] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding. Yolov10: Real-time end-to-end object detection, 2024.

[18] D. Werner, A. Al-Hamadi, and P. Werner. Truncated signed distance function: Experiments on voxel size. In A. Campilho and M. Kamel, editors, *Image Analysis and Recognition*, pages 357–364, Cham, 2014. Springer International Publishing.