2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Osiris: Building Hierarchical Representations for Agricultural Environments

Adam Mukuddem[1], Paul Amayo[2]

***Abstract*— 3D scene graphs have recently emerged as a powerful and human-understandable way of representing complex 3D environments. These describe environments through a layered or hierarchical graph where nodes represent different spatial concepts (from low-level geometry to higher-level scene-scale reasoning) and the edges between them represent relationships. While these representations have shown great promise in indoor well-structured environments, their use in outdoor structured environments such as agricultural environments has been under-explored. A key challenge here is that concepts and structures often observed in urban indoor environments cannot be easily transferred to these novel scenes.**

Motivated by this challenge, this paper presents *Osiris* which is a 3D scene graph builder for agricultural environments. We first propose a structure of the hierarchical graph for agricultural environments consisting of rowed crops and through our proposed system *Osiris* incrementally construct a 3D scene graph of agricultural environments from data taken onboard a mobile robot. We validate and evaluate the performance of Osiris using real-world data collected at several farms and show that this system is able to accurately get to the underlying structure of these agricultural environments while presenting a metrically accurate and human-understandable representation.

I. Introduction

As robotic technology has continued to advance over the last few years, there has been an increase in the deployment of robots that are not only navigating around different environments but performing specialised tasks. One area that has seen an increase in the deployment of such robots has been the agricultural sector, where robots have been used for pesticide deployment, fruit picking, and smart farming practices [1]. The success of each of these robots at these tasks is however closely tied to how well the robot can understand these agricultural environments and maintain a metrically accurate map of the observed scenes [2].

At the same time, there has been emerging global interest in cooperative robotics in agriculture [3], whereby humans and robots work in unison to perform these specialised tasks, increasing their individual productivity and that of the entire sector. However, a key ingredient needed to maximise cooperative performance is to ensure there is a high level of human-robot coordination. One of several methods to achieve this is to create a mutual understanding of environmental interpretation. In this way, robots embed an understanding of the ways in which humans interpret the environment. With the resulting outputs from the robot being human-understandable, coordination between the two could be greatly boosted and ultimately allow for robots to understand natural language instructions given by humans.

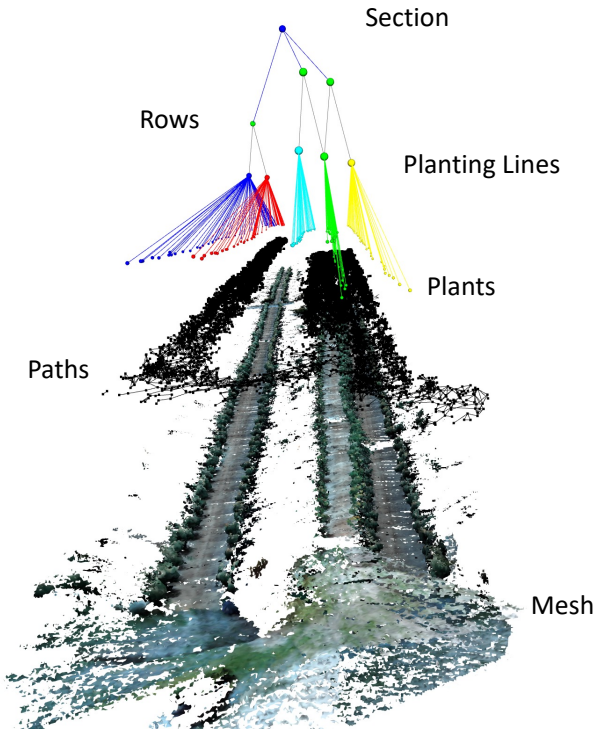

Fig. 1. We present Osiris, which is system that builds 3D Scene Graphs for agricultural environments from sensor data. The figure shows a 3D scene graph created by Osiris of a berry farm.

The creation of robotic environmental representations that are not only metrically accurate, to ensure that a robot can perform its tasks safely and accurately, and at the same time human-understandable, to allow for higher levels of coordination is not trivial. 3D scene graphs have however recently emerged as a powerful high-level representation of 3D environments integrating geometry and semantics at multiple levels of abstraction [4]. A 3D scene graph is a hierarchical graph where nodes represent spatial concepts at varying layers of abstraction and edges represent relationships between concepts [4]. There has been significant development in scene graph generation, however, existing techniques are restricted to only a few types of abstractions (rooms, buildings) and are mostly limited to indoor scenes.

[1]Adam Mukuddem (mkdada001@myuct.ac.za) and [2]Paul Amayo (paul.amayo@uct.ac.za) are with the African Robotics Unit, University of Cape Town, South Africa.

In this work we present *Osiris* which is a 3D Scene Graph construction system for outdoor agricultural environments. Despite their outdoor nature many agricultural environments, particularly those involved with the growing of rowed crops, are highly structured. By leveraging this structure, *Osiris* is able to build a hierarchical representation of this environment with varying layers of abstraction as the robot moves through the environment. The layers of *Osiris* are sections, rows, planting lines, plants, paths, and a 3D metric semantic mesh. The choice of layers is based on the structure of the row planting system and is designed with task and motion planning queries in mind.

This system affords robots working in agricultural environments a closer to 'human-like' interpretation of agricultural environments while still ensuring their task performance is not degraded. In this way creating a path for these robots to understand and execute natural language instructions from humans but also allowing for planning to occur at higher levels of abstractions and not just at the low-level geometry. In both ways increasing the productivity of the deployed robots.

We summarise the main contributions in this paper as follows:

- We introduce the concept of a 3D scene graph for agricultural environments.
- We develop *Osiris*, which is a 3D Scene Graph construction system for agricultural environments.
- We validate and evaluate the performance of *Osiris*, using real-world data collected in agricultural environments.

The paper is organized as follows. In Section II related work is discussed. In Section III, we present an overview of *Osiris*. Section IV showcases the performance of *Osiris* on real-world farm data collected. Finally, conclusions are drawn in Section V.

II. Related Work

Armeni *et al.* [5] introduced the notion of 3D scene graphs as a hierarchical model of 3D environments. The 3D Scene graph in [5] was used to ground geometric and semantic information into the representation of an indoor environment and to portray relationships between semantic concepts. In ([6], [7]), the 3D scene graph from [5] was extended such that the 3D scene graph was built directly from sensor data. Hughes *et al.* [8] extended the work in [6] and [7] to develop Hydra, which is a real-time 3D scene generation and optimization system built from sensor data. Hydra [8] was predominantly developed for and tested in indoor environments. The layers of the scene graph in Hydra [8] (from high to low) are buildings, rooms, places, objects, and a metric-semantic 3D mesh. Bavle et al. [9], [10] were able to combine a 3D scene graph formulation obtained from architectural plans of a building and tightly couple it with the SLAM state of the robot and jointly optimise them creating situational graphs or S-Graphs. The S-graphs have recently been deployed for the online localisation of a legged robot [11] with performance comparable to other state-of-the-art pipelines displayed, albeit in only indoor environments. Thus the gap remains in creating and utilising 3D scene graphs for the outdoor agricultural use-case.

III. Agricultural 3D Scene Graphs

This section introduces the concept of a 3D Scene graph for agricultural environments and its incremental construction from image data collected by a mobile robot. Most agricultural environments are outdoors and have a structure to them. Structure in agriculture relates to the spatial arrangement of crops within a growing area [12]. Row planting is one of the many agricultural planting systems [12]. Crops such as grapes, berries, potatoes, and sugar beets are planted in a row system. Row planting has a structure to it where crops are planted in lines. This introduces the idea of planting lines as a semantic concept related to the row planting system. We based the layers of *Osiris*, on the structure of the row planting system while at the same time considering the different task and motion queries robots in agricultural environments encounter.

The layers of *Osiris* are thus farm sections, robot-navigable rows, planting lines, plants, paths, and a 3D metric semantic mesh. *Layer 1* is a metric-semantic 3D mesh. *Layer 2* is a graph of paths within the farm. Paths represent connections between free space in the farm and form a topological map of the environment which can be used for navigation planning queries. *Layer 3* is subgraph of the plant objects detected within the farm. *Layer 4* is a subgraph of the planting lines. Planting lines represent the lines in which the crops are planted. *Layer 5* is a subgraph of the rows within the farm. In this work, we define a row as a robot-navigable area located in between two planting lines, in this way two planting lines form the lower layer of a specific row. *Layer 6* is a subgraph of the sections within the farm. Sections are areas of the farm for which the crops are of a specific crop, as a farm can contain multiple crops.

Formally we define the hierarchical graph obtained through *Osiris* as $\mathcal{O} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges between nodes $\mathcal{V}$. The set of nodes $\mathcal{V}$ can be partitioned into the six layers of the graph as follows. $\mathcal{V} = \cup_{i=1}^{6} \mathcal{V}^i$. Figure 2 illustrates the process used by *Osiris* to obtain the graph $\mathcal{O}$. This is described in more detail in the following sections.

A. Metric Semantic 3D Mesh Construction

The base layer of the Osiris scene-graph is a 3D Metric-Semantic Mesh obtained in real-time from a stream of image data. To enhance performance and ensure that the resulting system is robot-deployable, a key-frame based approach was taken for the construction as can be seen in Figure 2. This involves firstly taking the image stream being fed into *Osiris* from a stereo camera and obtaining the ego-motion estimate of the camera. Within this work, ego-motion was obtained using ORBSLAM3 [13]. Following a significant change in motion, a new key-frame is added which condenses the new information in the scene.

With key-frames obtained their depth and semantic labels are then sought. To obtain depth a bespoke lightweight network based on PyDNet2 [14] and trained on agricultural

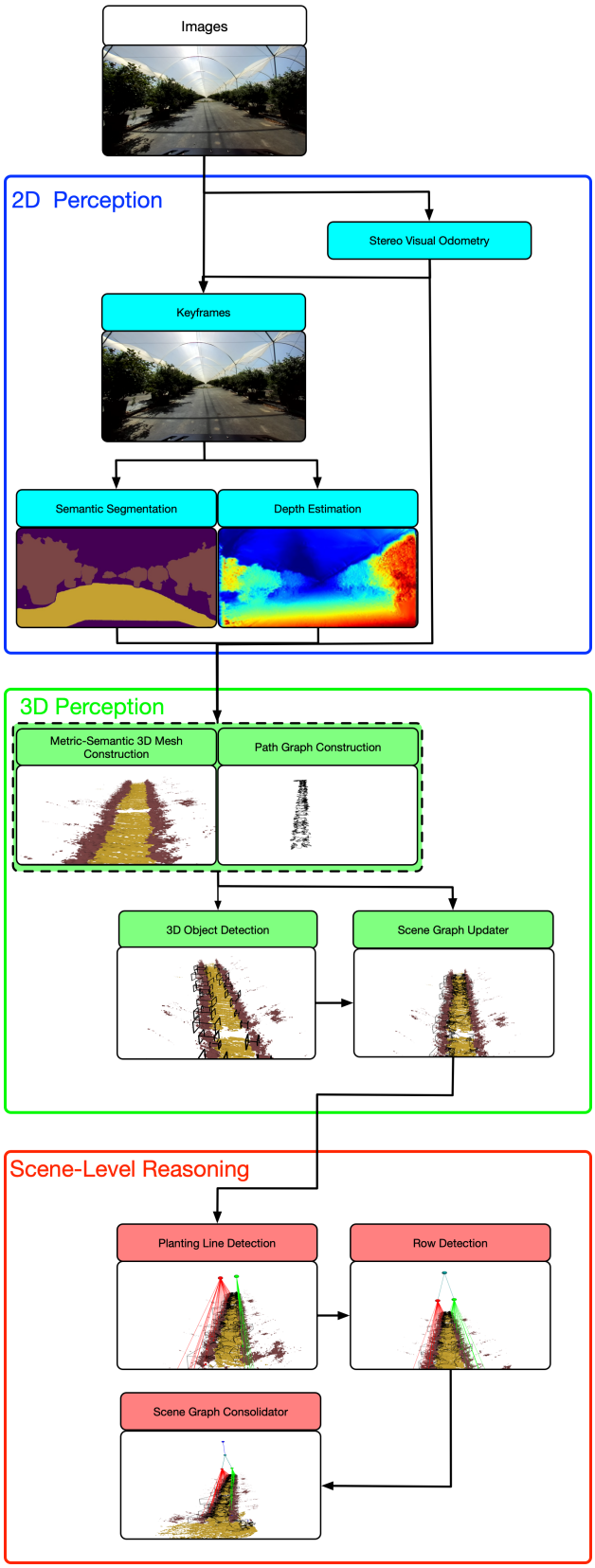

Fig. 2. Figure illustrating the process flow of *Osiris*. From a stream of RGB images obtained from a camera *Osiris* is able to create a hierarchical scene graph by performing 2D then 3D perception followed by reasoning over the entire scene.

field data is used to estimate depth. This network can run in real-time on a CPU while still maintaining accurate depth.

For the semantic segmentation, a two-pronged approach was used. This consisted of prompt-guided bounding box detection followed by high-quality semantic segmentation within these bounding boxes. The objective behind this is that agricultural environments reveal a wide range of objects and subsequently semantic classes with many of these not contributing directly to the underlying agricultural tasks. At the same time there is a large variation in the ways rowed crops appear in the scene, be they through shrubs, stand-alone bushes or fruits on contiguous trees. By leveraging a prompt-guided approach *Osiris* is able to generalise to more diverse ways in which rowed crops are grown. Allowing this system to represent them in a semantic mesh while also ensuring that objects that are not used in the agricultural tasks do not persist within the graph.

To obtain the bounding boxes the approach presented by Grounding DINO [15] method was utilised. A set of keywords corresponding to classes of agricultural produce (‘fruits’,’plants’,‘bushes‘) were fed into this approach alongside a further keyword ‘paths’ to capture the traversable area observed by the robot. The approach from Segment Anything [16] was then taken which used the bounding boxes produced by Grounding DINO [15] as guides for semantic segmentation.

The resulting Semantic segmentation and depth estimation were used as inputs to produce a semantically labeled pointcloud as shown in Figure 2. Voxblox [17] is then used to integrate the semantic pointcloud into a Truncated Signed Distance Field (TSDF) and a Euclidean Distance Field (ESDF), as in Hydra [8] only information within a specified radius of the robot is used to create the TSDF and ESDF. The 3D metric semantic mesh is then extracted using Voxblox’s [17] marching cube implementation. While performing the marching cubes voxels that correspond to a zero-crossing in the TSDF i.e. those that contain a surface are retrieved and labelled. It is from these voxels that the paths in *Layer 2* are constructed. The construction of the paths is detailed in the following section.

B. Path Graph Construction

To obtain paths, a similar approach to that taken in Hydra [8] is employed where a sub-graph is extracted using the Generalised Voronoi diagram (GVD), which is built during the ESDF integration in *Layer 1*. The GVD is a set of voxels that are equidistant to at least two obstacles. Obstacles in this instance are the afore-mentioned voxels corresponding to surfaces. The number of obstacles a voxel is equidistant to are now referred to as its basis points.

The GVD thus forms an exact description of the free space within an environment [18], and thus reveals obstacle-free robot navigable paths within the scene, making it invaluable for planning. However, it typically contains a large number of voxels which makes it difficult to manipulate. To make it more amenable for robot tasks, we incrementally sparsify the GVD using the approach in [19]. In this, a subset of GVD voxels are extracted as nodes and edges if they have 3 or more basis points. If a voxel has 4 or more basis points or

if the neighborhood of a voxel matches a template proposed in [19], the voxel is considered a path node $v_i^2 = (x_i, y_i, z_i)$ and added to the set of nodes $\mathcal{V}^2$ corresponding to this layer which are then sent to the scene graph updater. Here x_i, y_i, z_i represents the 3D position of the voxel.

Edges between nodes are added in two phases. First, we label voxels with 3 basis points with the ID of the path node it is closest to and add edges between all neighboring path nodes. Second, we analyze all the edges added, by calculating the distance between the new straight-line edges and each of the GVD voxels between the place nodes. If the distance exceeds a deviation threshold set, the voxel with the greatest distance to the edge is added as a path node and the initial straight line edge between the two path nodes is split. We iterate over these two phases several times at every keyframe. If there are any disconnected nodes after several iterations of the two phases at every keyframe, they are removed from the subgraph. This then results in a sparsified graph consisting of path nodes and edges connecting the nodes which represent traversability.

These path nodes however still remain intricately linked to the surface voxels revealed in *Layer 1*. This allows our system to link path nodes v^2 with the closest vertex of the 3D mesh related to the surface voxels. In this way edges between *Layer 1* and *Layer 2* are created. These vertex nodes $v_i^1 = (x_i, y_i, z_i)$ are also then added to the set of nodes $\mathcal{V}^1$ corresponding to the mesh layer which are then sent to the scene graph updater. Here x_i, y_i, z_i represents the position of the vertex.

C. 3D Object Detection

To obtain *Layer 3* as can be seen in Figure 2, object detection on the 3D metric-semantic mesh generated in *Layer 1* is performed. The object detection is obtained through Euclidean clustering [20] of mesh vertices that were updated on the addition of the new keyframe. A key variable used in Euclidean clustering [20] is *cluster tolerance*, which relates to the minimum allowed distance between vertices. Vertices are joined together if they are less than *cluster tolerance* apart. An adequate value of this is determined through experiments performed on the robot.

Euclidean clustering is additionally further restricted to semantic classes corresponding to agricultural produce as described in the construction of *Layer 1*. Thus only objects belonging to plants are obtained and propagated through the scene graph. The clustering reveals a cluster of vertices as well as a bounding box and its centroid for each object detected. For each object obtained the centroid position $v^3 = (x_i, y_i, z_i)$ is used to create the object node and is added to the set of nodes $\mathcal{V}^3$ corresponding to the object layer. At the same time, an edge is added between the bounding box centroid and the clustered vertices of the detected object. Thus an additional vertex node $v_i^1 = (x_i, y_i, z_i)$ is added to the set of nodes $\mathcal{V}^1$ corresponding to the mesh layer for each object detected which are then sent to the scene graph updater.

D. Scene Graph Updater

The scene graph updater receives the new set of nodes $\mathcal{V} = \mathcal{V}^1, \mathcal{V}^2, \mathcal{V}^3$ obtained at every key frame corresponding to the first three layers of the scene graph together with their respective edges and incrementally updates the graph $\mathcal{O}$. After being updated the entire scene graph can now be reasoned over. This reasoning is described in the following sections.

E. Planting Lines Detection

The detection of planting lines is the first task in the Osiris pipeline that reasons at the entire scene level. This is as it considers the entirety of objects in a scene and not just those introduced by the latest keyframe. This is crucial as the system seeks to extract the underlying structure of row-crop systems. Planting lines are therefore generated from the object nodes $\mathcal{V}^3$ belonging to *Layer 3*.

To further pronounce the structure of row-system crops a 2D birds eye view of the plant objects is obtained as a pre-processing step. This reveals the view of the objects from above which both pronounces this underlying structure and eases detection of the planting lines but also significantly reduces the problem to a 2D multi-line estimation problem. In doing so a reduced set, $\mathcal{P} = [p_1, p_2, \cdots, p_{N_o}]$ is created where N_o is the number of plant objects, $p_i = [x_i, y_i]$ and x_i and y_i are the coordinates of the plant objects found in the original set $\mathcal{V}^3$.

With the problem reduced to the fitting of multiple geometric line models to the set of plants $\mathcal{P}$, the Convex Relaxation Algorithm (CORAL) [21] can be used. The CORAL algorithm is a method for fitting multiple geometric models to multi-structured data via convex relaxation [21]. The result of the CORAL algorithm is a set of labels $\mathcal{L}$, which assigns each plant object in the set $\mathcal{P}$ to a geometric line model $\mathcal{M} = [M_1, M_2, \cdots, M_{N_l}]$ where M_i is the 2D equation of a line and N_l is the number of labels.

CORAL approaches the multi-model estimation as an energy minimisation problem. In this way an energy functional that aims for a solution that is geometrically accurate, spatially smooth as well as compact is created as below:

$$E(\mathcal{L}) = \sum_{l=1}^{N_l} \int_{\Omega} \rho_l(\mathcal{P}, \phi_l(\mathcal{P}))d\Omega + \lambda \sum_{l=1}^{N_l} \int_{\Omega} \omega_N R(\nabla_N \phi_l(\mathcal{P}))d\Omega + \beta N_l \quad (1)$$

The first term of this energy functional represents the geometric cost of assigning a plant object to particular line model. Here $\phi_l(\mathcal{P})$ represents the label currently assigned to the object. The residual function $\rho_l(\mathcal{P}, \phi_l(P))$ gives the euclidean distance between the plant object's position and the geometric line model M_l. It therefore penalises labels assignments that poorly correspond to the underlying data.

The second term of the energy functional represents a smoothness cost. We expect that in rowed crops plants that are physically close together have a high probability of belonging to the same planting line. The function R is formulated penalises neighbouring plant objects that do not

share the same label. This is calculated by leveraging ∇_N which calculates the gradient of the current label assignment between neighbouring plant objects. λ trades-off between the smoothness and the geometric cost while the weighting function ω_N allows for finer control of the influence of neighbouring points, in this case the weighting is reduced the further a plants neighbour is. N here is the number of neighbours that every plant object will have.

The last term promotes compactnesss by favouring label assignments that explain the data with as few planting lines models as possible, this reduces redundant models and ensures that the solution more closely resembles the underlying structure. The value β trades off the compactness against smoothness and geometric cost.

The optimal label assignment $\mathcal{L}$ can be obtained through the minimisation of Equation 1. The CORAL algorithm [21] is able to simultaneously minimise over the geometric, smoothness, and compactness term. This presents a solution that is geometrically sound, spatially smooth and without redundant labels. Additionally due to its highly, parallel nature, CORAL's time performance does not reduce significantly even as the number of plant objects within the scene N_o is increasing. This allows it to reason over large scenes without incurring a large time penalty.

Planting lines are therefore detected through CORAL on all plant objects within the scene after every keyframe has been processed. The corresponding labelling $\mathcal{L}$ presents edges between the plant object nodes $\mathcal{V}^3$ and the line models $\mathcal{M}$ which create line model nodes. Line model nodes $\upsilon_i^4 = (x_i, y_i, z_i)$ are thus added to the set of nodes $\mathcal{V}^4$ and added to the scene graph $\mathcal{O}$. To obtain (x_i, y_i, z_i) the mean of all the positions of plant objects nodes that are assigned to the corresponding line model is used.

F. Row Detection

Within this work, rows are defined to be traversable sections that give access to the planting lines and subsequently the planting objects. This follows quite naturally from the definition of paths found in *Layer 2* and we expect that most of the place nodes will lie within rows. Rows therefore occur in between planting lines and the geometric line models $\mathcal{M}$ obtained while extracting the plant line nodes in *Layer 4* can be used for the row detection.

For a given pair of line models, M_i and M_j, a gradient check is first performed to ensure they are within a tolerance threshold $t_{||}$ of being parallel. Subsequently, the distance between the two planting lines is computed and if it lies between specified min and max tolerance values guided by the physical farm structure then a row is added between the two lines. A row node $\upsilon_i^5 = (x_i, y_i, z_i)$ is thus added to the set of nodes $\mathcal{V}^5$ corresponding to the row layer and added to the scene graph $\mathcal{O}$. Edges between this node υ^5 and the two plant line nodes (υ_i^4,υ_j^4) are also added to the graph. To obtain (x_i, y_i, z_i) the mean of all the positions of plant line nodes that are assigned to the row node are used.

It is important to note that in this way, a plant line node υ^4 can belong more than one row node ($\upsilon_i^5, \upsilon_i^5$). This varies from most definitions of hierarchical graphs where most current approaches ensure that each node at a lower layer can only have a single edge to a node in the layer above it [4], [8], [18], [7]. Enforcing this would create a level of duplication when dealing with shared spatial concepts which does not faithfully represent the underlying structure. By retaining the idea of shared spatial concepts in a similar manner to that seen in S-graphs [9], [10] whereby 'walls' can be shared by 'rooms', *Osiris* is able to retain a high-fidelity representation of the underlying environment.

G. Scene Graph Consolidator

The scene graph consolidator then creates the final layer by consolidating the row nodes $\mathcal{V}^5$ into a section node $\upsilon_i^6 = (x_i, y_i, z_i)$ that is added to the set of nodes $\mathcal{V}^5$ corresponding to the section layer and the scene graph $\mathcal{O}$. Edges between this node and the row nodes are also added to the graph.

IV. Experimental Evaluation

We present our experiments to demonstrate the capabilities of *Osiris*. We used real-world data collected from a berry farm in South Africa to test *Osiris*.

A. Data

We collected data using a front-facing Zed 2I stereo camera on board a wheeled robot platform, Clearthpath Robotic's Husky A200 [22]. We recorded data with a 2.1mm lens, at 15 frames per second, 720p, with a field of view of 120 degrees. The husky was driven up and down several rows the each farm. The berry farm is approximately 36 000 m^2 and consists of 14 rows. Each row is approximately 250m long and we captured 3 rows of the farm. Two of the rows were adjacent and the last row had a one-row gap between the other two. The husky was also equipped with a Velodyne HDL-32E LiDAR scanner operating at 10Hz whose calibration to the camera was known. Data collected from the LiDAR was used to create a ground-truth pointcloud from which we could probe the performance of *Osiris*.

B. Experiments

To measure the performance of *Osiris* we present an extensive evaluation of the accuracy and runtime performance of the system. We calculated various metrics for each layer of *Osiris*. We created ground truth data by manually segmenting plant objects from the ground-truth LiDAR pointcloud data. Bounding boxes were manually placed over the plant objects by a subject-expert labeler using a custom Graphical User Interface (GUI). Plant objects were then manually aggregated into planting lines and planting lines into rows.

*1) **Accuracy evaluation: Plant Objects**:* To report on the accuracy of the plant object detection. Similar metrics to those presented in [8] were calculated. These were the percentage of objects in the ground truth annotated pointcloud that have an object estimated by *Osiris* with the correct semantic label within a specified radius ("% Found") and the percentage of objects in the estimated scene graph that have a ground-truth object with the correct semantic label within a specified radius ("% Correct"). Results for this were shown in Figure 3 and obtained over a selection of different cluster tolerances used for the Euclidean Clustering.

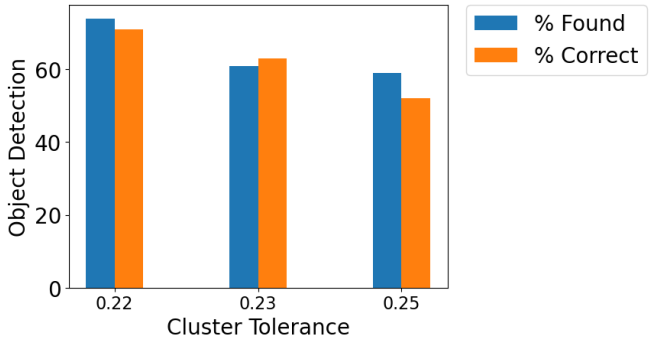

Fig. 3. Osiris performance on object detection in *Layer 3* using different cluster tolerances (m) and averaged over three independent runs.

As can be seen in Figure 3 *Osiris* maintained a high percentage of objects detected with a cluster tolerance of 0.22. Increasing this leading to a slight decrease in performance. Attention then shifted to the performance over planting line detection.

*2) **Accuracy evaluation: Planting Lines**:* To evaluate the detection of the planting lines, rather than comparing the obtained geometric models, our evaluation sought to determine whether *Osiris* over or under segments the scene into planting lines. Precision and recall metric defined in [23] for room detection were thus adapted to this task as below.

$$Precision = \frac{1}{|\mathcal{L}_e|} \sum_{l_e \in \mathcal{L}_e} \max_{l_g \in \mathcal{L}_g} \frac{|l_g \cap l_e|}{|l_e|}$$
$$Recall = \frac{1}{|\mathcal{L}_g|} \sum_{l_g \in \mathcal{L}_g} \max_{l_e \in \mathcal{L}_e} \frac{|l_e \cap l_g|}{|l_g|} \quad (2)$$

Here $\mathcal{L}_e$ is the set of estimated lines, $\mathcal{L}_g$ is the set of ground-truth lines, and each line l_e or l_g is defined as a set of voxels that correspond to the area of the planting lines. This in the case of the ground truth is manually annotated but for the evaluation of *Osiris* this can be determined by the planting objects assigned to each line model. From the results presented in Figure 4 it can be seen that our system maintains both high precision and recall at optimal parameters. If the compactness term β of CORAL is further increased the performance of the line detection decreases as can be expected, increasing β encourages a very compact solution which reduces the number of planting lines that *Osiris* estimates beyond that in the data.

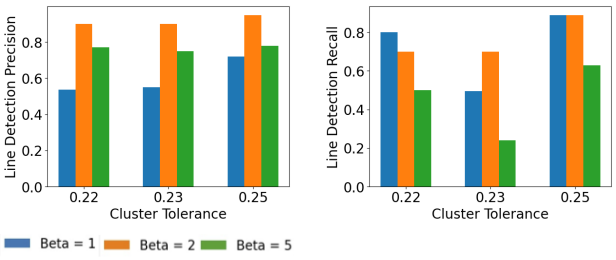

Fig. 4. Osiris performance on line detection in *Layer 4* using different cluster tolerances (m) and compactness terms averaged over three independent runs.

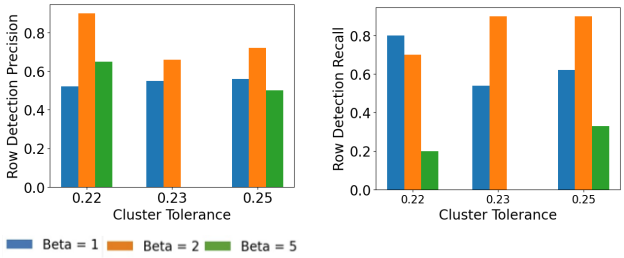

Fig. 5. Osiris performance on row detection in *Layer 5* using different cluster tolerances (m) and averaged over three runs

*3) **Accuracy evaluation: Rows**:* The row detection performance was then evaluated, using the same procedure as defined for the line detection with results presented in Figure 5. As was the case in the line detection good performance of both precision and recall were reported for the most optimal parameters. There was then a noticeable drop particularly in the recall of the rows when the compactness paramter β was increased further as can be expected.

V. Conclusion

This paper introduces a novel notion of a 3D Scene Graph for the representation of agricultural environments and then presents *Osiris* which is a system that can incrementally build this scene graph using only a stream of camera data from a mobile robot. While there have been many systems capable of constructing scene graphs these have been mostly deployed in indoor environments whose structure greatly differs from that of agricultural environments over which rowed crops are grown.

We thus define the layers of *Osiris* as farm sections, navigable rows, planting lines, plant objects, places and a 3D metric semantic mesh. When applied to data collected from agricultural regions our proposed system showed that it is able to incrementally construct a scene graph with good performance shown in obtaining the underlying structure of the agricultural environment. This provides a path for robots deployed in agricultural environments to harness the benefits that dynamic scene graphs offer. These can be used to boost robotic prediction, planning, decision-making and human coordination in these environments.

Acknowledgments

The authors would like to thank the Back Family for the gracious access to their farm.

References

[1] R. Sparrow and M. Howard, "Robots in agriculture: prospects, impacts, ethics, and policy," *precision agriculture*, vol. 22, pp. 818–833, 2021.
[2] A. J. Davison, "Futuremapping: The computational structure of spatial ai systems," *arXiv preprint arXiv:1803.11288*, 2018.
[3] C. Lytridis, V. G. Kaburlasos, T. Pachidis, M. Manios, E. Vrochidou, T. Kalampokas, and S. Chatzistamatis, "An overview of cooperative robotics in agriculture," *Agronomy*, vol. 11, no. 9, p. 1818, 2021.
[4] J. Strader, W. Chen, N. Hughes, A. Speranzon, and L. Carlone, "Informing 3d scene graph generation with common-sense spatial knowledge."
[5] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3d scene graph: A structure for unified semantics, 3d space, and camera," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5664–5673.

[6] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans," *arXiv preprint arXiv:2002.06289*, 2020.

[7] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, "Kimera: From slam to spatial perception with 3d dynamic scene graphs," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1510–1546, 2021.

[8] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," 2022.

[9] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "Situational graphs for robot navigation in structured indoor environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9107–9114, 2022.

[10] ——, "S-graphs+: Real-time localization and mapping leveraging hierarchical representations," *arXiv preprint arXiv:2212.11770*, 2022.

[11] M. Shaheer, J. A. Millan-Romera, H. Bavle, J. L. Sanchez-Lopez, J. Civera, and H. Voos, "Graph-based global robot localization informing situational graphs with architectural graphs," *arXiv preprint arXiv:2303.02076*, 2023.

[12] E. H. Satorre, *Spatial Crop Structurespatial crop structurein Agricultural Systems*. New York, NY: Springer New York, 2013, pp. 1513–1528. [Online]. Available: https://doi.org/10.1007/978-1-4614-5797-8_223

[13] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, 2021.

[14] M. Poggi, F. Tosi, F. Aleotti, and S. Mattoccia, "Real-time self-supervised monocular depth estimation without gpu," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[15] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, and L. Zhang, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," 2023.

[16] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023.

[17] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1366–1373.

[18] N. Hughes, Y. Chang, S. Hu, R. Talak, R. Abdulhai, J. Strader, and L. Carlone, "Foundations of spatial perception for robotics: Hierarchical representations and real-time systems," *arXiv preprint arXiv:2305.07154*, 2023.

[19] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Sparse 3d topological graphs for micro-aerial vehicle planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[20] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1–4.

[21] P. Amayo, P. Piniés, L. M. Paz, and P. Newman, "Geometric multi-model fitting with a convex relaxation algorithm," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8138–8146.

[22] C. Robotics. (n.d.) Husky unmanned ground vehicle robot. Retrieved on 2024-03-04. [Online]. Available: https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/

[23] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, "Room segmentation: Survey, implementation, and analysis," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1019–1026.