# Multi-Hypothesis Tracking in a Graph-Based World Model for Knowledge-Driven Active Perception

Jordy Senden, Kevin Hollands, David Rapado-Rincon, Akshay Kumar Burusa, Bas Herremans, Herman Bruyninckx, and René van de Molengraft

***Abstract*—Robots that have to robustly execute their task in an environment containing many variations need situational awareness to adapt at run-time. This work proposes a knowledge-centered software architecture with a world model (WM) as a first class citizen, from which other software components can query information in order to infer predictions, configure skills, and monitor the progress of the task. This approach is demonstrated on the task of detecting tomato trusses hanging from a plant, with possible occlusions from leaves. A labeled property graph is used to model a tomato plant, which can be queried to create predictions of truss locations. This information is used to configure two tomato detection skills. First the plant is passively scanned for trusses. Association of the obtained information to the semantic objects in the model leads to multiple semantic hypotheses, that are explicitly modeled in the graph world model. If trusses are missing according to a hypothesis the second skill actively looks at inferred position of the undetected trusses. Tests shows that this approach of context-aware active perception allows the robot to decide when to look for missing trusses, which improves the detection of occluded trusses. Moreover, by keeping the task-, skill-, and semantic association functionalities agnostic to the context, but relying on the answers to the queries to the world model, the approach is composable and flexible. This is shown by a qualitative test on a different tomato plant.**

***Index Terms*—World models, robotics, reasoning, multiple hypothesis tracking, MHT, active perception.**

## I. Introduction

A DECLINING human labor-force in the agro-food sector drives the need for robotic systems to automate repetitive tasks, such as harvesting fruits and pruning trees. Automation is challenging since typical agro-food environments contain many objects or actors, like humans, animals, plants or even other robots, which can exhibit variations in both their appearance and behavior. For example, the natural growth of tomato trees in greenhouses will lead to (visual) disparities and dense foliage cause many trusses to be visually occluded, as can be seen from Fig. 1, which makes it difficult for a robot to detect, localize and handle them. The goal is to create autonomous systems that can robustly execute their task in semi-structured environments, like harvesting tomato trusses in a greenhouse. Current applications are moslty data-driven, where information is only derived from raw sensor data and the decisions on when, where and how to move are decided at design time. Such purely data-driven approaches lack flexibility to adapt to the variations in these environments [1]. Instead of merely improving detection algorithms, it is important to understand the context in which detection algorithms are deployed and accept imperfect, noisy data as a matter of fact and incorporating it into the overall processing strategy, which was already argued in [2]. The main challenge is to model this context, i.e. the knowledge about the robot and its environment, explicitly and provide it to the robot. This knowledge, captured in a so-called world model (WM), should drive the planning, action, performance evaluation, and

(a) (b)

Fig. 1. Picture inside a tomato greenhouse, showing the dense environment (a) and visual occlusion of tomatoes (b).

Manuscript received 16 March 2023; accepted 12 July 2023. Date of publication 31 July 2023; date of current version 10 August 2023. This letter was recommended for publication by Associate Editor E. Ugur and Editor T. Ogata upon evaluation of the reviewers' comments. This work was part of the project Cognitive Robotics for Flexible Agro-food Technology (FlexCRAFT), supported by the Netherlands Organisation for Scientific Research (NWO) under Grant P17-01. *(Corresponding author: Jordy Patrick Franciscus Senden.)*

Jordy Senden, Kevin Hollands, Bas Herremans, and René van de Molengraft are with the Department of Mechanical Engineering, TU Eindhoven, 5612AZ Eindhoven, The Netherlands (e-mail: j.p.f.senden@tue.nl; k.hollands@ziggo.nl; bas.herremans@gmail.com; m.j.g.v.d.molengraft@tue.nl).

David Rapado-Rincon and Akshay Kumar Burusa are with the Farm Technology Group, Wageningen University and Research, 6700AA Wageningen, The Netherlands (e-mail: david.rapadorincon@wur.nl; akshaykumar.burusa@wur.nl).

Herman Bruyninckx is with the Department of Mechanical Engineering, TU Eindhoven, 5612AZ Eindhoven, The Netherlands, also with the Department of Mechanical Engineering, KU Leuven, B-3001 Leuven, Belgium, and also with Flanders Make, B-3001 Leuven, Belgium (e-mail: herman.bruyninckx@kuleuven.be).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3300282, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3300282

data-association efforts of the robot. The robot must evaluate the situation they are in, based on the current state of the world model, and monitor the task in order to actively re-plan an re-configure their actions when necessary.

## II. Related Work

For applications like autonomous mobile robots (AMR) the WM is often a geometric map, which can be provided a-priori or created by the robot itself using Simultaneous Localization and Mapping (SLAM) [3]. New sensor data is compared to the geometric map for localization and object tracking, e.g. using particle filtering [4]. Many hypotheses are created that compare a fictive measurement in the WM to the newly measured real data. Geometric maps, including such metric hypotheses, scale poorly with the size of the environment, the dimensions of the map, and the number of degrees-of-freedom (DOFs) of the robot with respect to this map. Therefore, often a probabilistic approach is taken, where the least probable hypotheses, which could possibly contain the correct one, are discarded. Expanding on pure geometric models, [5] suggest a semantic, object centered world model. When new measurements are done, they have to be anchored to semantic objects in this WM. Although implicit, [6] shows that adding knowledge of relations between object, i.e. how connected objects move together, can improve tracking performance and reduces the number of possible hypotheses. In [7] an explicit, geometric Building Information Model (BIM) is used to extract relevant features that can be detected with a Laser Range Finder (LRF). Known recurring patterns between these features in the model and the LRF measurements are used to reduce the number of hypotheses for localization. Using active perception [8], [9] the robot can gather more information to resolve the multiple hypotheses into a single conclusive belief state, i.e. create situational awareness. In [10] it is shown that introducing prior knowledge in the form of geometric Regions of Interest (ROIs) improves the performance of Next-Best-View planning (NBV) for active perception for tomato plant reconstruction. However, all these approaches are rooted or heavily rely on exact geometric WMs which lack the capacity to explicitly model semantic connections between objects. In semi-structured environments, such as a greenhouse, the geometry of the environment might contain many variations. For robust task execution, the environment should be described on a level of abstraction that contains invariant features [11], e.g. relative distances or sizes, equality in shape or reflection, or a known topology.

## III. Contributions

The objective is to develop a method for a robot to localize desired objects, and understand when it is missing information and where to find it. Our first contribution is to develop an explicit object-centered world model that is able to model different types of semantic relations between objects, adding to the work in [10]. Secondly, we provide the robot with a mechanism that can query information from the WM, or infer information through reasoning, which is used for planning and skill-configuration. A graph structure is chosen to represent high-level concepts and relations on the symbolical level, which is driven by multi-robot cooperation and robust task execution in a world containing variations [12]. The approach in [7] is reversed; a knowledge graph is created where geometric information can be added. The graph structure enables context-driven graph traversals to retrieve information and to infer facts that are not explicitly modeled. Our third contribution is the association of new information to the WM. This happens at the semantic level, based on the explicit connections in the WM, which will result in semantic hypothesis. The semantic association relies on object properties as well as the relations between them, which can be a strong feature in itself. Therefore, association is done only after enough information is acquired, instead of filtering every new measurement. Our fourth contribution is to track the resulting multiple semantic hypotheses in the graph-based world model. This allows us to track, test and resolve these semantic hypotheses through active perception and update the WM with newly measured information. We show this on a robot that is tasked with detecting all tomato trusses on a single plant, while being faced with occlusions.

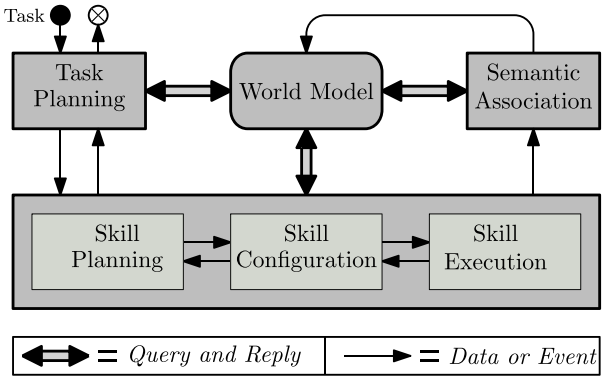



Fig. 2. Software overview with the world model at the center. The other components interact with the WM through queries.

## IV. Method

The world model must be created separately from its intended use, i.e. agnostic to *how* it is going to be used, and become an explicit component in the software architecture of the system. As a first class citizen, the WM can be used by other software components that can query information from it. Using tomato-truss harvesting as an example, this section shows how prior knowledge of the environment is modeled, how the robot uses this knowledge to configure its detection skills and how new detections lead to multiple semantic hypotheses when they are associated with known objects in the WM.

### A. Software Architecture

The software architecture in Fig. 2 shows that the central world model is surrounded by three functional components; a task interpretation block (left), a skill block (bottom) and a semantic association block (right). The skill block consists of three sub-functionalities; planning, configuration and execution. These three functional components interact with the world

model through queries, indicated with the thick double arrows. The task block interprets a given assignment and decides on a task-plan, conforming to the current belief state of the world. With one or more possible task-plans, the system needs to choose the correct skills to perform this plan. The internal parameters of the skill need be to configure dependent on the current information of the world. After a skill is executed the information gained or changes made in the environment need to be updated in the world model. This new information needs to be associated with information that already exists in the world model. This work will not focus on task planning or skill selection, but rather on the world model, information queries and semantic association.

### B. World Model

Because the WM is designed independent from its intended use, there is no knowledge on where data is stored or which messages are passed along. Instead of following a publish-subscribe messaging pattern, the functional component need to request or infer information through queries to the WM. The Labeled Property Graph (LPG) is chosen as a meta-model for the WM to fulfill these requirements.

*1) Labeled Property Graph:* A labeled property graph is a type of graph database that provides support for highly interconnected information. An LPG consists of vertices $V$, which are connected by edges $E$ that make up the graph $G(V, E)$. In a robotics context vertices can be used among others to represent physical objects. In the LPG model the vertices and edges can have properties. These properties should describe the physical characteristics of the object itself, e.g. shape, reflection, or mass, and can only change by physically changing the object. Edges can model multiple different semantic relations between objects, e.g. mereological relations like *is_part_of*, physical relations like *connected_to*, or geometric relations like *distance_between*. The properties of these graph-edges capture the relevant values that describe the connection, like the value of a distance. Some edge properties rely on the properties of the connected objects, e.g. the friction-coefficient *between* objects relies on the surface roughness properties of the individual objects. When an edge property is dependent on a vertex that it is not connected to, the edge has to be extended with a help-vertex, since the LPG model does not allow to connect edges to vertices or other edges. An example is the *geometric_connection*, which describes the (relative) pose of an object as a vector, expressed with respect to a reference frame. In the graph WM this connection is modeled as a set of three edges and one extra help-vertex, as shown in Fig. 3. The reference frame is modeled as a vertex. Two edges model the direction of the vector and the third points to the reference frame in which it is expressed. The extra vertex holds the values of the vector. This model makes it possible to describe an objects position in multiple different reference frames at the same time and to ask the WM which objects are modeled in the same frame. There is no absolute world frame, only relative frames that are connected. When the robot starts moving, only the relation between its own frame and connected surrounding frames needs to be updated. The relative position of the trusses, which are expressed in the plant frame, do not change when the robot moves. Updating frames and resolving contradictions when a position is expressed in multiple frames is out of the scope of this work.

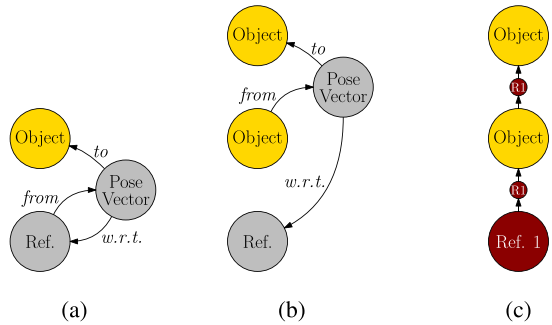



Fig. 3. Example of the *geometric_connection*, which describes the pose of an object as a vector, expressed with respect to a reference frame. This pose can be absolute w.r.t. the reference frame (a) or relative to another object (b). In the remainder of this work, the *geometric_connection* is visualized as (c).

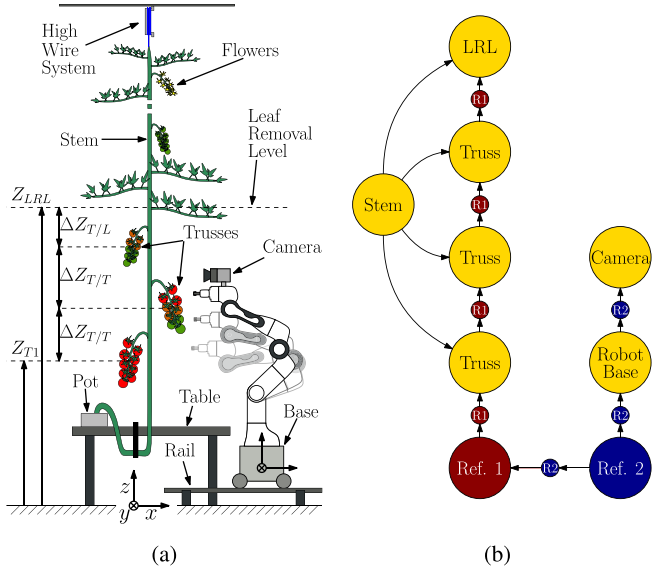



Fig. 4. Visualization of a tomato plant in a greenhouse (a) and the graph model representation (b).

*2) LPG of a Tomato Plant:* The world model for a tomato harvesting robot must contain knowledge of the tomato plants inside a greenhouse, schematically shown in Fig. 4(a). These plants continue to grow throughout the season, while the lower leaves of the plant are pruned for better air circulation, light penetration, and accessibility to ripe trusses. The grower closely monitors and controls the environment, which results in a consistent growth and equality across plants, such as tomato size, amount of tomatoes per truss and inter-truss distance, which are strong invariant features. Although all tomato plants are geometrically different, their topology and invariant features hold for all plants in the greenhouse. Modeling the world on the abstraction level of these invariants can increase robustness in task execution [11]. In Fig. 4(b) the graph model for a tomato plant is shown, where the topology is capture by *physical_connection* edges and the geometric connections are simplified for improved visualization.

```
graph = Graph()
g = graph.traversal()
ID_stem = g.addV("Object").property("name","Stem")
    .next()
ID_truss = g.addV("Object").property("name","Truss")
    .next()
g.addE("physical_connection").from_(ID_stem)
    .to(ID_truss).next()
```

Listing 1. Gremlin-Python example where a graph is created.

```
ID_list = g.V(ID_stem).both("physical_connection")
    .hasLabel("Object").has("name","Truss").toList()
path_list = g.V(ID_list)
    .repeat(__.both("Geometric_con").simplePath())
    .until(__.hasLabel("Reference")).path()
    .by("name").toList()
```

Listing 2. Gremlin-Python example where the graph is queried for information.

### C. Querying the World Model

The graph is created using the Apache TinkerPop graph computing framework[1]. Gremlin[2] is the graph traversal language of Apache TinkerPop, which is used to write queries to interact with the LPG. Gremlin-Python is used to implement Gremlin within the Python language, by connecting to a server that is hosting the graph [13]. In Listing 1 a piece of example code is shown where a graph is initialized. The graph-traversal source $g$ is used to start queries to create the graph. Two vertices are created that represent an *Object*, one a stem and the other a truss, which are connected by an edge with the label *physical_connection*. Queries for information use the same graph-traversal source $g$. The example in Listing 2 first asks for a list of IDs of all trusses that are physically connected to the stem. Next, for all objects in this list their geometric position in the world is queried. The traverser will start at one of the *object* vertices and traverse over *Geometric_con* edges until a *Reference* vertex is reached. The vertices that are encountered during this traversal are returned by their *name* in a list.

### D. Task

A task is a desired change in the environment, or information gain thereof, and should be described agnostic to the system that is going to perform it. Hard-coding recipes that couple actions to predicted situations, such as a finite state machine (FSM), is not robust against variation. Instead, the system should ask the right questions to the WM and act accordingly based on the answers. These questions should be captured in a query machine (QM), analogous to a FSM, which consists of a set of task-related questions, e.g. "what information do I need to collect in the world", "do I posses a skill that can perform this measurement?", or "do I have the correct information to execute this skill?". Transitioning between these questions in the QM depends on the received answers that do or do not satisfy certain criteria. This work will not focus on parsing of a high-level task or researching properties of such query machine. Instead, the query machine is created explicitly for the task and thus knows some of the context. The simplified task in this work is: "for each tomato plant, localize all tomato trusses below the leaf removal level". The system is provided with the model discussed in Section IV-B2 and only needs to retrieve information from the environment, not physically interact with it. The task-plan is captured in a simplified query machine, shown in Algorithm 1.

**Algorithm 1:** Query Machine - Taskplan.

```
TaskDone = False
SkillsAvailable = True
T ← query for truss IDs
L ← query for leaf-deck IDs
while ¬TaskDone & SkillsAvailable do
    [T_a^H, H] ← query for all T_i in T associated with
      a measurement, return hypothesis H
    if (T_a^H == T & ||H|| == 1) then
        TaskDone = True
    else
        SkillsAvailable ← execute skillplan
```

Here *TaskDone* and *SkillsAvailable* are boolean flags that keep track on whether the task is finished and if there are still skills available to continue with the task. The truss IDs are stored in list $\boldsymbol{T}$ and the leaf-deck ID is stored in $L$. Trusses ($T_i$) that are associated with a measurement are stored in list $\boldsymbol{T}_a^{\boldsymbol{H}}$ and $H$ represents the hypotheses they belong to.

### E. Skill-Plan

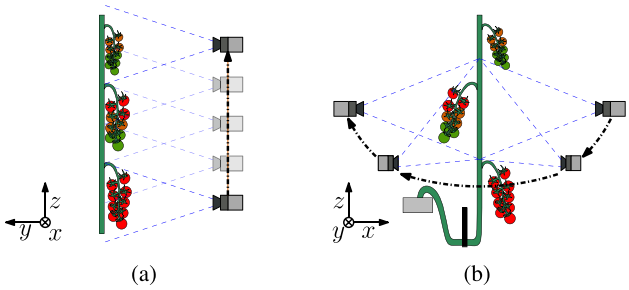
Localizing the trusses is done with a depth camera that is attached to the end-effector (EE) of a robot arm. The basic sub-skills that are used in this work are:

1) move the depth camera relative to the tomato plant
2) localize single tomatoes using a neural network
3) cluster detected tomatoes into a truss

The details of the detection- and clustering skills, including pre- and post-processing of the data, are explained in [14]. When these skills would be configured once at design time, a choice has to be made on how to scan the plant to find all trusses below the leaf-removal level (LRL). At least, the plant has to be scanned along the stem, ranging from the lowest possible truss position until the maximum height of the LRL. Only scanning it from one side requires few viewpoints, but has a high change of missing occluded trusses. Alternatively, more viewpoints can be chosen that view the plant from different sides. This approach might detect an occluded truss by looking from another side, but taking more viewpoints costs time. In [10] it is already shown that an attention mechanism can reduce the number of viewpoints. We use a similar approach of knowledge driven active perception. Two skills are designed, which are visualized in Fig. 5 , that search for trusses using different operation modes. The current information in the WM, either based on prior knowledge or recent measurements, determines which skill to perform and how.

Skill A scans the plant for multiple trusses by moving the camera upward along the stem. The scanning range is a configurable

[1][Online]. Available: https://tinkerpop.apache.org
[2][Online]. Available: https://tinkerpop.apache.org/gremlin.html




(a) (b)

Fig. 5. Visualization of detection skills. Skill A passively scans the plant upward along the stem for multiple trusses, based on prior knowledge an open-loop guarded motion (a), skill B actively looks for an expected truss at a specific position by scanning in a circular motion around the plant (b).

**Algorithm 2:** Query Machine - Skillplan.

**query** for $(\boldsymbol{T}, \boldsymbol{T}_a^H, \boldsymbol{H}, L)$
**if** $\boldsymbol{T}_a^H = \emptyset$ **then**
  $\boldsymbol{Z}_r^A \leftarrow$ **calculate** ROI($\boldsymbol{T}$,$L$)
  $[\boldsymbol{M}, \boldsymbol{E_A}] \leftarrow$ **perform** Skill A in $\boldsymbol{Z}_r^A$
  **return** measurements $\boldsymbol{M}$ and skill-events $\boldsymbol{E_A}$
**else**
  **forall** *hypotheses* $H_j$ *in* $\boldsymbol{H}$ **do**
    $\boldsymbol{T}_a^{H_j} \leftarrow$ **query** all $T_i$ that are associated with a measurement according to $H_j$?
    **forall** $T_i$ *in* $(\boldsymbol{T} - \boldsymbol{T}_a^{H_j})$ **do**
      $\boldsymbol{Z}_r^B \leftarrow$ **calculate** ROI($\boldsymbol{T}_a^{H_j}, \boldsymbol{T}, L$)
      $[\boldsymbol{M}, \boldsymbol{E}_B] \leftarrow$ **perform** Skill B in $\boldsymbol{Z}_r^B$
    **return** measurements $\boldsymbol{M}$ and skill-events $\boldsymbol{E}_B$

skill-parameter, which is determined by the lowest predicted truss and the maximum height of the predicted LRL. Skill B scans around the plant for a single truss by moving the camera in a circular motion around a center point. The center point is a skill-parameter that is determined by the center position of an expected truss. Skill composition, i.e. determining a skill-plan depending on the situation and on what the skills can provide, is not the focus of this work. Instead, a fixed skill-plan is created in the form of a query-machine, similar to the task-plan, as shown in Algorithm 2.

When no trusses are detected yet ($\boldsymbol{T}_a^H = \emptyset$), skill A is deployed to try and find multiple trusses. A region of interest (ROI) for skill A to scan is calculated based on the a-priori estimated tomato locations ($\boldsymbol{T}$) and LRL ($L$), which are provided to the robot at the start of its task. Measurements are stored in $\boldsymbol{M}$ and skill-event ($\boldsymbol{E_A}$) are sent to the task-planner. Skill B is deployed after some trusses are measured or if skill A failed to detect trusses due to possible occlusions. For each resulting hypothesis $H_j$, the ROI for skill B is determined by predicting the locations of trusses in the WM ($\boldsymbol{T}$) based on the position of the measured trusses $\boldsymbol{T}_a^{H_j}$ and the LRL ($L$). Where skill A passively scans the plant based on prior knowledge an open-loop guarded motion, skill B actively looks for a specific truss. Measurements lead to multiple hypotheses as to which trusses are measured and thus where other trusses could be, depending on a specific hypothesis. In this work all hypotheses are equally probable to prevent the urge of pruning the least probable hypothesis. Instead hypotheses are actively tested, by deploying skill B, to gather new information in order to reach a conclusion if possible. This pipeline of gathering information, updating the WM and actively looking for missing information is what we call active perception.

### F. Semantic Association

Each new measurement with the depth camera can lead to newly detected tomatoes. The position of these tomatoes are tracked and filtered over time, and eventually clustered in a truss, which is explained in [14]. After these data-association steps result in sufficiently likely semantic objects, i.e. a truss, they have to be associated with the information in the WM.

*1) Feature and Relation Matching:* Association is done on semantic features of the object itself or on relations between objects. This is very similar to [15] with the difference that the initial semantic map $G_m$ in this example is modeled based on prior knowledge, instead of being a result of previous measurements. To be able to associate on connections, and when a WM update is not required instantly, it is better to first detect multiple objects to compare to the model. This allows for a smoothing approach instead of filtering [16] which is more resistance against noise and outliers. After scanning for multiple objects they can be matched to the WM as a set. The goal of the matching algorithm is to determine which objects in the set of detections ($\boldsymbol{S_d}$) belong to the set of predicted objects ($\boldsymbol{S_m}$) in the WM. The matching algorithm queries the WM for prior information about the objects of interest and the relations between them. If the set of predicted objects ($\boldsymbol{S_m}$) are directly (geometrically) connected to each other, this connection is first used for the association. If there is no relative connection between objects, or if only a single object is measured, the absolute pose with respect to their frame of origin is used. This can lead to many hypotheses, because the absolute height-range of a single modeled trusses can overlap. The geometric relations are modeled in the reference frame of the plant, while the truss positions are measured in a robot frame. To match the measurements to the model, a mapping is necessary to express the (relative) geometric relations in the same frame. The current implementation assumes the prior knowledge in the WM is correct. If there are more measurements than modeled object, it is assumed that $\boldsymbol{S_d}$ contains outliers caused by false positive measurements or trusses from another plant. To deal with this, one or more detections are removed from $\boldsymbol{S_d}$ and the remaining detections are matched to $\boldsymbol{S_m}$ until a feasible hypothesis is found.

*2) Multi-Hypotheses Tracking in the Graph:* Often, a set of measurements cannot be unambiguously matched with the set of predictions. Instead, the many matching possibilities lead to multiple hypotheses, which should be tracked in a MHT approach. Each hypothesis should be tested by deploying skill B to search for missing trusses, until a single hypothesis is considered to be true. Since the objects in the prediction set $S_m$ can have semantic relations, each hypothesis should be able to link this prediction set to a set of measured objects. Moreover, hypotheses that arise from a single association should be linked

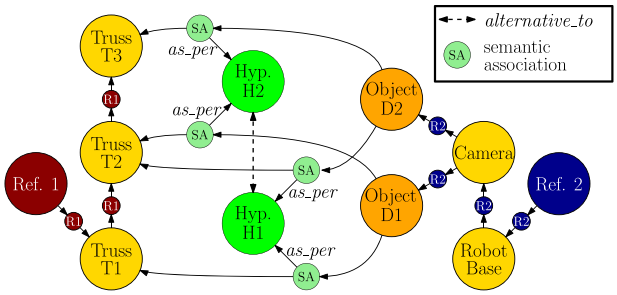



Fig. 6. Example of MHT in a graph. Hypothesis-vertices connect a set of new data to objects in the WM. Hypothesis H1 associates D1 with T1 and D2 with T2, while alternative hypothesis H2 associates D1 with T2 and D2 with T3.


Fig. 7. Test setup showing the ABB robot and mock-up plant (a). The occluded truss can be detected from the side (b). Example videos of situations 3 and 4 can be viewed by scanning the QR-codes (c) and (d), respectively.

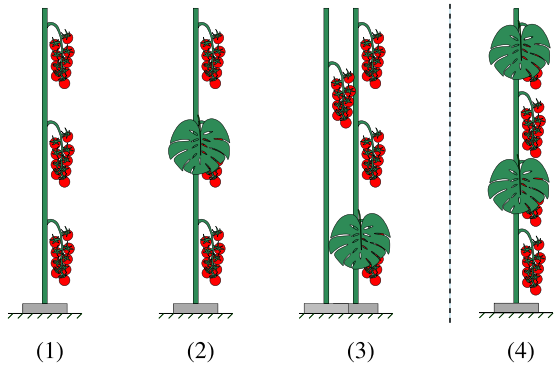



Fig. 8. Overview of tested situations, representing different realistic variations in a plant with three trusses: (1) without occlusions, (2) with occlusions, (3) with unexpected 'extra' truss. Situation (4) shows a different plant with four trusses, which will (qualitatively) show that, by only changing the prior knowledge in the WM, our approach still works.

together to enable first order logic between them; "if hypothesis A is true, B must be false". To keep track of all hypotheses, and how they are related, the concept of a hypothesis is introduced as vertex in the graph, which is visualized in Fig. 6. Measurement vertices are connected with to their associated object vertex in the WM with an *semantic_association* edge. Because a set of measurements could be associated to different objects, according to different hypotheses, the *semantic_association* edges should connect to the hypothesis vertex they belong to. This is similar to a *geometric_connection* that points to the reference it is expressed in (Fig. 3). In our work, the hypothesis vertex and association help-vertex do no contain properties. Explicitly introducing hypothesis-vertices allows queries like: "which objects are detected according to hypothesis X?", "what are alternatives to hypothesis X?", etc. When one hypothesis is believed to be true, the set of measurements belonging to that hypothesis is anchored to the predicted object in the WM and the alternative hypotheses can be removed from the graph. Another benefit of explicitly modeling hypothesis in the graph is the ability to share them with other robots by sharing the WM.

## V. Experiments and Results

An experimental mock-up tomato plant is created, which allows control over the variations that are introduced in the environment. This way, the robot's task output can be compared to a known ground truth of the environment, which is necessary to validate the framework against variations and occlusions. This section will explain the test setup, explain the different types of situations that are tested and discuss the performance of our approach.

### A. Setup

An Intel RealSense L515 LiDAR camera is attached to the EE of an ABB IRB1200 robotic arm. The robot operating system (ROS) is used to simplify communication between different processes running on the robot. MoveIt[3], a motion planning framework for ROS, is used for motion planning and control of the arm. The LiDAR camera is calibrated by performing an eye-in-hand calibration and the experiments are performed indoors, where the lighting conditions are controlled. A mock-up of a tomato plant is created, shown in Fig. 7(a), where fake trusses are connected to a vertical pole that represents the stem. The number of trusses and height of these trusses along the stem can be varied and occluding leaves can be introduced, shown in Fig. 7(b). The robot is placed in front of the mock-up plant at a distance of approximately one meter. This value is chosen to be within the field-of-view of the camera, while still enabling the robot to have a decent motion range while scanning.

[3][Online]. Available: https://moveit.ros.org

### B. Situations

To determine the performance of the approach against variations in the environment, several situations that are observed in a real greenhouse are emulated on the mock-up plant. In Fig. 8 a schematic overview of these situations is shown. In all situations the height of each truss is chosen from a set of measurements that are performed in a real greenhouse and scaled down to fit the parameters of the test setup. This ensures that the distribution of the variation in inter-truss distance is realistic. For situations 1 to 3, the plant consists of three trusses that are attached to the stem. In situation 1 there are no occlusions and all trusses should be visible while performing skill A. In this situation only the height of the trusses is varied between tests. In situation 2 trusses are occluded by a leaf; in situation 2.1 only the lowest truss is occluded, in situation 2.2 only the middle truss is

occluded, and in situation 2.3 two random trusses are occluded. The difference between 2.1 and 2.2 is the possible hypotheses that the matching algorithm can make; seeing two trusses closely together results in different hypotheses than seeing two trusses spaced apart. Situation 2.3 is tested to see how the algorithm fares with multiple occlusions and only one visible truss, which prevents association on the relative connections between trusses. In situation 3 an extra truss is introduced, which represents a truss from a neighboring plant; in situation 3.1 the 'extra' truss is hanging close to one of the other trusses, in situation 3.2 the 'extra' truss is positioned in between two trusses, and in situation 3.3 the 'extra' truss is randomly added while one of the three existing trusses is occluded. Since this extra truss or plant is not part of the world model, there will be issues associating the measurement to the model. Where situation 1 to 3 are designed to quantify the performance while facing realistic variations, situation 4 tests the generality of the approach. In situation 4 a truss is added to the mock-up, representing a different type of tomato tree that has 4 trusses below the leaf removal deck. The graph-based model, as discussed in IV-B2, is extended to include an extra truss and the geometric relations are updated accordingly. This situation qualitatively shows that the task- and skill-plan as well as the matching algorithm is agnostic to the WM, but their operations are based on the answers to the set of queries. In the video that can be viewed by scanning the QR-code in Fig. 7(c), an example experiment of situation 2.1 shows a visualization of the graph-traversals and an explanation of what the robot is doing at each step. Note that at the start the WM contains a graph of prior knowledge, which is queried for information to configure skill A. After semantic association, the queries traverse the measured values to predict the position of undetected trusses to configure skill B.

### C. Performance

To quantify the performance of the approach the position of the detected trusses, in situations 1 to 3, is compared to the known ground-truth of the mock-up plant. If a detection is made within a distance of 6 cm from the ground-truth, a value based on the size of the used trusses, it is deemed as correct. These detections lead to hypotheses which are not always resolved into a single conclusion. A second quantitative performance indicator is the number of times the approach comes up with the right conclusion or a set of hypotheses that contain the correct conclusion. Situation 4 give a qualitatively indication of the flexibility of the approach. A video showcasing situation 4 can be seen when scanning the QR-code in Fig. 7(d).

*1) Results Precision and Recall:* To quantify the added value of knowledge-driven deployement of skill B to actively test hypotheses when a truss is believed to be occluded, the precision and recall after performing both skills is determined for situations 1 to 3. Precision is a measure for the amount of false-positive detections, while recall indicates how much of the trusses are actually detected. The results are shown in Table I, which show the amount of tests for each situation, the gain of performance after deploying skill B and the number of viewpoints taken. Since the detection algorithm is the same for both skills, the precision remains equal. A slight drop in precision-gain for is caused by executing skill B on top of skill A. Doing more measurements increase the chance of wrong detections. In situations with occlusions the number of viewpoints increase. The robot understands that it is missing information and deploys skill B, which increases the recall. The link between precision and recall is caused by the clustering approach, because if two trusses hang closely together they can be clustered into one truss, resulting in a false positive. The association algorithm matches it with a truss in the WM and moves on without recognizing it missed the two real trusses. This leads to two false negatives on top of the false positive.

TABLE I
EXPERIMENTAL RESULT FOR SITUATIONS 1 TO 3

| **Situation** | 1 | 2.1 | 2.2 | 2.3 | 3.1 | 3.2 | 3.3 |
|---|---|---|---|---|---|---|---|
| Nr. of tests | 60 | 20 | 20 | 20 | 20 | 20 | 20 |
| **Precision** | | | | | | | |
| Skill A | 97 | 98 | 100 | 91 | 90 | 90 | 98 |
| Skill A&B | 97 | 98 | 98 | 86 | 90 | 90 | 99 |
| Gain | 0 | 0 | -2 | -5 | 0 | 0 | 1 |
| **Recall** | | | | | | | |
| Skill A | 97 | 67 | 67 | 33 | 88 | 86 | 75 |
| Skill A&B | 97 | 97 | 90 | 85 | 88 | 86 | 90 |
| Gain | 0 | 30 | 23 | 52 | 0 | 0 | 15 |
| **Number of viewpoints** | | | | | | | |
| Mean ($\mu$) | 5.6 | 7.6 | 7 | 9.1 | 5.4 | 5 | 7.3 |
| Std ($\sigma$) | 0.8 | 1.3 | 0 | 1.1 | 1.6 | 0 | 2.1 |

The precision, recall and number of viewpoints is shown after deploying skill a and after skill B. In situations with occlusions, the robot decides to search further, which improves the recall.

*2) Results Hypotheses Correctness:* Since the robot's own perception of performance and its skill-plan are based on the hypotheses, their correctness is important. When the task is finished, there are three final outcomes:

- True: the robot has seen all objects that it was expecting and decides correctly that its task is finished.
- Undetermined: the task is not yet finished but there are no other skills to deploy to test the unresolved hypotheses. However, the correct conclusion is present in this set.
- False: the robot has drawn a wrong conclusion or is left with a set of hypotheses which are all false.

The results, after both skills are deployed in their respective ROIs, are shown in Fig. 9 . It shows that in situation 1 either correct or false conclusions are drawn. All expected trusses are seen but sometimes clustered wrongly, which will result in a wrong conclusion, there are no hypotheses left. In situation 2 the level of incorrect conclusions is linked to the precision of the detection algorithm. In situation 2.2 some hypotheses could not be resolved into the correct conclusion because there was often a problem of detecting the truss behind the occlusion, as seen from the recall in Table I. In situation 3 it was very difficult to draw the correct conclusion. Since the WM has no knowledge of 'extra trusses' it is impossible to draw a single correct conclusion. Instead, it tries to fit all trusses on the one

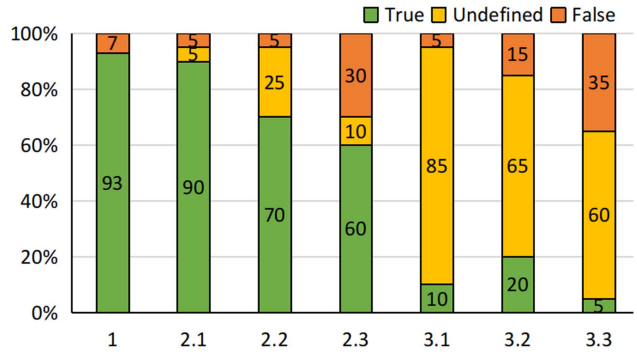



Fig. 9. Conclusion after task execution per situations.

modeled plant, which results in multiple hypotheses that often contain the correct conclusion.

## VI. Conclusion

A method is created that enables a robot to localize trusses on a plant, and understand when it misses expected information. The world model at the center of the software architecture stores the information, be it prior knowledge or measured detection, which is used by the other functional components of the system. The graph structure allows to model objects and their semantic relations. Gathering a number of measurements before semantic matching allows us to take these relations into account, which help to reduce the number of hypotheses. The set of hypotheses are explicitly modeled as vertices in the graph, which allows to query for missing information according to each hypothesis. This in turn can help the robot decide whether or not to deploy a second skill that actively looks for this missing information. Linking hypotheses to each other allows for first order logic on the hypothesis-level, which is used to reach a final conclusion. The current implementation assumes that the information is the world model is correct. In the future, we need a mechanism that enables to add missing information based on trustworthy measurements. Other future steps could be to make a smarter choice on which hypothesis to test first, e.g. by modeling a set of hypotheses as a Bayesian network, assigning probabilities to each hypothesis. This can help the robot to decide which hypothesis to test first; the one that requires the least amount of effort, while maximizing the information gain. Although the approach is demonstrated on a tomato harvesting use-case, we believe it is relevant in other robotic contexts where the environment is semi-structured.

## Acknowledgment

This project is supported by NWO/TTW. Special thanks to Vereijken Kwekerijen for welcoming us to their greenhouse.

## References

[1] A. L. Yuille and C. Liu, "Deep Nets: What have they ever done for vision?," *Int. J. Comput. Vis.*, vol. 129, no. 3, pp. 781–802, 2021, doi: 10.1007/s11263-020-01405-z.

[2] R. Bajcsy, "Active perception," *Proc. IEEE*, vol. 76, no. 8, pp. 966–1005, Aug. 1988.

[3] D. M. Rosen, K. J. Doherty, A. Terán Espinoza, and J. J. Leonard, "Advances in inference and representation for simultaneous localization and mapping," *Annu. Rev. Control Robot., Auton. Syst.*, vol. 4, no. 1, pp. 215–242, May 2021.

[4] N. Bore, J. Ekekrantz, P. Jensfelt, and J. Folkesson, "Detection and tracking of general movable objects in large 3D maps," 2017, *arXiv:1712.08409*.

[5] J. Elfring, S. Van Den Dries, M. J. G. van de Molengraft, and M. Steinbuch, "Semantic world modeling using probabilistic multiple hypothesis anchoring," *Robot. Auton. Syst.*, vol. 61, no. 2, pp. 95–105, 2013, doi: 10.1016/j.robot.2012.11.005.

[6] W. Houtman et al., "Automated flower counting from partial detections: Multiple hypothesis tracking with a connected-flower plant model," *Comput. Electron. Agriculture*, vol. 188, 2021, Art. no. 106346. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016816992100363X

[7] R. W. M. Hendrikx, P. Pauwels, E. Torta, H. Bruyninckx, and M. J. G. van de Molengraft, "Connecting semantic building information models and robotics: An application to 2D LiDAR-based localization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11654–11660.

[8] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Auton. Robots*, vol. 42, no. 2, pp. 177–196, Feb. 2018.

[9] Y. Zaky, G. Paruthi, B. Tripp, and J. Bergstra, "Active perception and representation for robotic manipulation," 2020, *arXiv:2003.06734*.

[10] A. K. Burusa, E. J. van Henten, and G. Kootstra, "Attention-driven active vision for efficient reconstruction of plants and targeted plant parts," 2022, *arXiv:2206.10274*.

[11] J. Senden, K. Jebbink, H. Bruyninckx, and M. J. G. van de Molengraft, "Invariant-based world models for robust robotic systems demonstrated on an autonomous football table," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 8542–8549, Jul. 2022.

[12] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artif. Intell.*, vol. 247, pp. 10–44, 2017, doi: 10.1016/j.artint.2014.11.003.

[13] K. R. Lawrence, "Practical Gremlin-An Apache Tinkerpop Tutorial," Version 283-preview, May 2022. [Online]. Available: https://www.kelvinlawrence.net/book/PracticalGremlin.pdf

[14] D. R. Rincon, E. J. van Henten, and G. Kootstra, "Development and evaluation of automated localization and reconstruction of all fruits on tomato plants in a greenhouse based on multi-view perception and 3D multi-object tracking," 2022, *arXiv:2211.02760*.

[15] G. Pramatarov, D. De Martini, M. Gadd, and P. Newman, "BoxGraph: Semantic place recognition and pose estimation from 3D LiDAR," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2022, pp. 7004–7011.

[16] A. Gelb, J. F. Kasper Jr., R. A. Nash Jr., C. F. Price, and A. A. Sutherland Jr., *Applied Optimal Estimation*, 16th ed., A. Gelb, Ed. Cambridge, MA, USA: The MIT Press, 2001.