3D Plant Structure Generation

Working towards an autonomous greenhouse

Daan Wijffels

Delft University of Technology

TU Delft

3D Plant Structure Generation

Working towards an autonomous greenhouse

by

Daan Wijffels

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday July 4, 2023.

Student number:	4476247	
Project duration:	April 22, 2022 – July 4, 2023	
Thesis committee:	Dr. ir. Y.B. Eisma,	TU Delft, supervisor
	Prof. dr. ir. J.C.F. de Winter,	TU Delft
	K. van den Berg,	Lely Technologies N.V.

ABSTRACT

As the global population continues to increase, so does its dependance on food, which has led to the development of more efficient food production methods. Greenhouse horticulture and genetically modified seeds have increased crop yield and decreased water and energy consumption. However, delicate greenhouse-grown vegetables continue to necessitate manual labour. The purpose of this thesis is to design and evaluate a vision system capable of detecting the spatial structure of greenhouse vegetation in order to automate plant repositioning, pruning, and leaf removal. The system employs a robotic arm and a single camera to generate custom plant features, match them, and generate a 3D model of the plant's structure. The thesis discusses the system's requirements, design, and evaluation, including the segmentation of plant outlines using deep learning networks, the construction of a 2D plant structure using constrained Delaunay triangulation, and triangulation-based depth calculation. Analysis of three-dimensional precision reveals a correlation between camera positional errors and the algorithm's depth estimation. The findings support the viability of using a vision system to automate greenhouse duties if certain modifications are made. Contributing to ongoing efforts to increase greenhouse agriculture's efficiency and reduce manual labour.

1 Introduction

According to Our World in Data [1], the current global population is increasing at 1.05 percent, or about 81 million additional individuals year over year. As a result, there is a constantly increasing demand for food, and thus the need to produce food as efficiently as possible. This efficiency can be measured in terms of water [2] or energy [3] consumption, as well as the amount of labour that is required [4]. We are now able to produce crops with a higher yield, all thanks to genetic modification of seeds [5]. Water usage can be decreased with to greenhouse horticulture [6], which has also made it possible to cultivate vegetables in previously impracticable regions [7]. And finally, mostly all of the world's field-grown vegetables, such as potatoes [8], corn [9], rice [10] and low-leaf vegetables like spinach [11] or cabbage [12], can be harvested semi-automatically, and thus require a lot less manual labour. As these crops either have firm fruit or are spread out widely in rows (in the case of low-leaf vegetables) they allow for an easier way of automating the harvest and crop care. However many other produce, especially those grown in a greenhouse, still require a great deal of human attention. These vegetables are more delicate, and are mostly farmed vertically when in a greenhouse. This is done to reduce the footprint, and maximize sunlight reaching the plants. Because of the smaller footprint, the energy usage of the greenhouse is reduced, as a smaller volume is heated. Figure 1 shows a row of tomatoes as they are typically grown in a greenhouse.

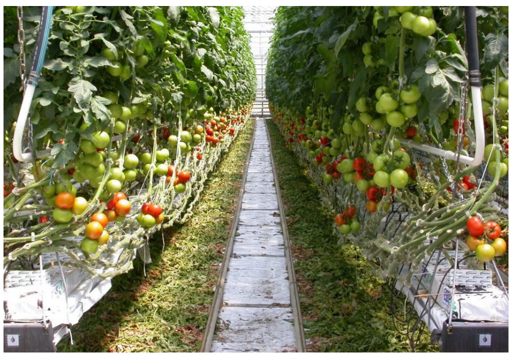


Fig. 1: A greenhouse row, with tomatoes grown vertically. [13]

Work in a greenhouse is challenging because of the high CO_2 concentration which increases plant growth but induces drowsiness in the workers [14], the warm temperatures especially during the summer months, and manual labour required. This is all for a low wage, which is determined by the price of the vegetables. As a result, there has been a push in recent years to automate the greenhouse. This also holds for Lely, the firm where this research was conducted. Lely is most recognized for their work in agriculture, specifically robotics for the dairy industry. Any device that replaces manual labour in a greenhouse should be capable of doing any, ideally all, of the functions now performed by workers: removal of low-hanging leaves as plants grow, pruning of suckers (small offshoots of the plant that hamper its growth), and specifically for tomato plants the repositioning of the plant as it grows. The repositioning is needed as the plants grow at around 0.25 meter per week, or about 13 meters in length in a year [15]. For this, the stem is wrapped around a small wire, as the plants grow the wire is elongated and moved horizontally at the top of the greenhouse. This allows the plant to sag as it grows, figure 2 shows a schematic of the tomato plants in a greenhouse. The repositioning, pruning of suckers, and removal of lower leaves has to be done every week. However, with current sizes of greenhouses this is a constant task for growers.

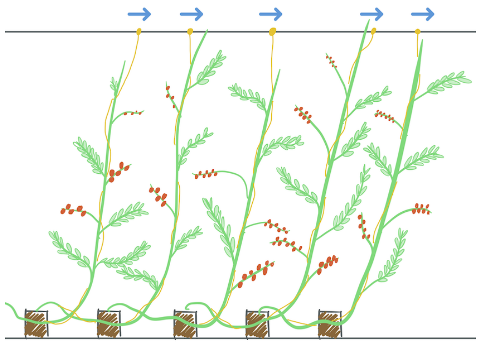


Fig. 2: Schematic visualization of tomato plant growth in a greenhouse, blue arrows showing the direction of repositioning, wrapping wires shown in yellow.

State-of-the-art robots as those from Wageningen University [16] and Chonnam National University [17] have proven to work. However, they are only designed to pick fruit, be that of sweet peppers or tomatoes in the cases mentioned. From this the robot from Wageningen University shows the most promise, however this comes at the cost of a large end effector which houses the vision system. The end effector size can become an obstacle when being used in more densely packed greenhouses, as is the case with tomato plants. In order to create a device that can replace most workers in a greenhouse, a system to detect the spatial structure of the vegetation could prove to be vital. As having an understanding of the plant's structure could allow for better detection and localization of plant features, such as: fruit, suckers, branches, and the stem. Which in turn could lead to solutions for solving the tasks mentioned above. A system is required to incorporate all these functions, while being small enough to manoeuver in between plants of the greenhouse. Thus culminating in the research aim of this thesis:

Designing and Evaluating a Vision System with the Goal of Detecting the Spatial Structure of Greenhouse Vegetation.

To accomplish this aim, the report has been structured as follows. The report starts with an overview of the previous work on the subject. By providing this background information, the current state of the art and the gaps that exist in the existing knowledge base are stated. Following this, the next two sections of the report focus on the requirements and design of the system. Section 3 provides a detailed description of the requirements of the robot, including the robot's design and the acquisition of input data. Section 4 discusses the algorithm's design, including an explanation of its building blocks. Once the design has been presented, the report moves onto an evaluation of the system in sections 5 and 6. Where the algorithm's accuracy and speed are tested and assessed. Ending with the discussion and conclusion of the report in sections 7 and 8.

2 Previous Work

As stated in the introduction, several research groups are actively working on greenhouse robotics. The majority of research is directed toward robotic tomato picking. Most likely because tomatoes are the most commonly grown vegetable in greenhouses [18], and the picking of tomatoes require the most workers. In this section an overview of other greenhouse robotics research is presented.

Several types of end-effectors have been created to grab tomatoes for picking. Chiu et al. where a combination of soft fingers and suction is used to remove tomatoes from the vine using a twisting motion [19], Gao et al. using two cups with small curved blades to cut the cherry tomato from its stem[20], and Jun et al. cutting tomatoes from the vine using a modified off-the-shelf scissor-like cutter are some of the studies being conducted in this field [17].

Next to this, full robotic systems to work in a greenhouse are also developed. These range from simple systems to spray pesticides on the crops [21], to fully automated picking robots like SWEEPER from Wageningen University [16]. From these, the latter shows the most promise of all cutting-edge robotics for use in a greenhouse. As it is the only one where development has progressed to the point where it can travel autonomously between plants and harvest fruit. SWEEPER, in contrast to most other studies, focuses on sweet peppers rather than tomatoes. It is based on a scissor lift system, which is commonly used by growers to move around the greenhouse. The end-effector uses a time-of-flight sensor as well as an RGB camera to detect the sweet peppers, in combination with a vibrating knife to cut the pepper stem. This all is attached to a 6-axis robotic arm in order to position the end-effector. With this design SWEEPER is able to pick 49% of all ripe sweet peppers in an average of 24 seconds per pepper. See figure 3 for an image of the SWEEPER robot.

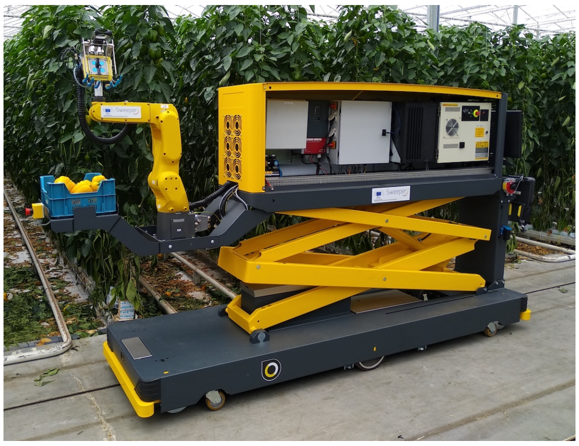


Fig. 3: The SWEEPER sweet pepper harvesting robot by Wageningen University [16], currently the most effective greenhouse harvesting robot.

All of the research mentioned above is for vegetable harvesting, however, as stated in the introduction, more tasks must be completed in order to fully automate a greenhouse. An understanding of the plant's spatial structure could provide the key to solving these tasks. This is referred to as the plant's structure in this thesis. Research has been done in the past to create such structures, however, most research focuses on the structure of trees. Neubert et al. uses the density of leaves in conjunction with a particle flow algorithm to recreate the tree in 3D [22]. Tan et al. uses a point cloud of a single tree as input and recreates the tree in 2 steps: first the visible branches (mostly the root of the tree) are created, followed by the occluded branches. The latter is recreated by using the location of leaves as an input [23]. Guo et al. again use a complete point cloud of a tree in order to recreate its structure [24]. This structure is constructed differently by limiting the angle between branch nodes and utilizing a model of tree stems and branches. By simulating the branch weight, it enhances the original structure. Next to these tree structure-generating papers is a paper focussing on plants from Quan et al. where each leaf is modeled independently and attached to the stem via a branch [25]. However, this is only shown to work on smaller plants with relatively large leaves.

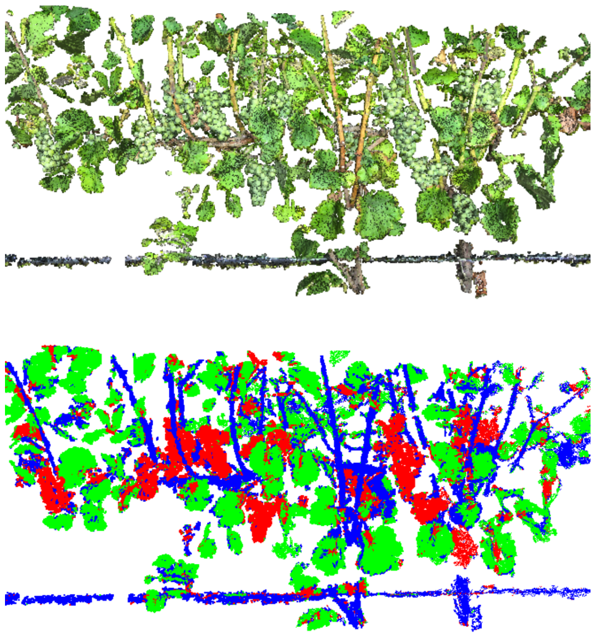


Fig. 4: Example of classification of vineyards as done by Dey et al. [26]. Grapes shown in red, leaves in green and plant stems shown in blue. From this example it can be seen that the wire at the bottom of the images is wrongly classified as a part of the plant stem.

Opposing these methods is a paper from Dey et al. [26], where point cloud data of a vineyard – in which plants are grown in a similar way to a greenhouse – is classified into: branches, grapes, and leaves. It does this by using a conditional random field algorithm and custom features representing the: point-ness, curve-ness, and surface-ness of a point in the point cloud based on its neighboring points. It is able to produce an incredible accuracy of 96% in its classification. See figure 4 for an example of the classification from this paper. Such an algorithm could be used to only select points that are part of branches and try to recreate a structure through these points. From examples in the paper it can be seen that this classification does make an error with wires – that are classified as branches– that are used in the vineyard. As the research was mostly focused on detecting the grapes, some alterations may have to be done to the algorithm to provide accurate input data on the branches.

As shown from the research above the gathering of point cloud data is essential for these algorithms to work. There are several ways to accomplish this: LiDAR, radar, stereovision, time-of-flight cameras, structure from motion, structured light, and the application of a deep learning algorithm are the most notable examples. A larger literature research, shown in appendix D, was carried out in order to select the optimum technique or sensor for usage in greenhouses. Every technique and sensor's operation was described in this study before being compared. A comparison was made on the following criteria: resolution, accuracy, speed, field of view, size, cost, and greenhouse usage. The last criterion, in contrast, was a subjective measure that took into account all potential barriers to a sensor's or technique's performance in a greenhouse. This analysis showed that stereovision offers the best method for obtaining depth information in a greenhouse setting.

3 Requirements

The studies above provide us with multiple routes for creating a plant structure, however they all fall short in some way. All studies that create a plant structure have in common that they use a point cloud of a single tree or plant in its entirety. Typically, a structure-from-motion system is used to create this data, using images from at least 120° surrounding the plant or tree. Unfortunately, it is virtually impossible to gather the same information in a greenhouse, as the plants are densely packed next to each other. This makes it extremely complex to determine which leaves –or in extensions which points in the point cloud– are connected to a single plant. Next to this, because of the layout of a typical greenhouse we can only view the plant from a specific angle, and from a short distance, making the constrained of at least 120° a challenging one. Even if all of these obstacles can be overcome, none of the methods provide a measurement of accuracy when comparing the 3D model to its real-world counterpart; they merely produce a representation of the plant. As a result, the structure-building methods described above would not be effective in a greenhouse environment. Allowing for this thesis to build upon these papers. The subsequent sections will examine: how a plant structure could aid with solving tasks in the greenhouse, the robot design utilized for this thesis, and ultimately how data is acquired for our algorithm.

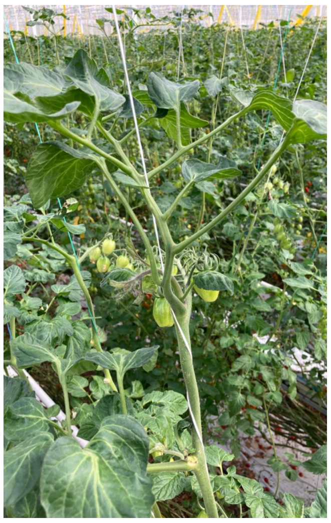
3.1 Greenhouse Tasks

As previously stated in this thesis; for a robotic system to replace most manual labour in the greenhouse 4 specific tasks – which take up most of the grower's time – have to be performed by the system. These tasks are as follows:

- Repositioning of plants with respect to its growth
- Removal of suckers
- Removal of low-hanging leaves
- Picking of fruit

All of these tasks are highly repetitive, therefore they show great potential to be automated. With the exception of fruit picking, getting an understanding of the spatial structure of the plant could provide vital in solving these tasks. This thesis defines a plant structure as follows: a list of linked 3D line segments representing the stem and, preferably, branches. In the following sections, we will illustrate how a plant structure can aid with the above-mentioned tasks.

Repositioning of the plants as they grow requires a thin string to be wrapped around the plant. By doing this the plant's weight is supported as it grows in a vertical fashion. Knowing the location of the plant's stem and branches is necessary to know where to place the string when wrapping it around the plant. See figure 5a for an example of this method. There does exist an alternative to this string method, called the Qlipr. The Qlipr is made by Pellikaan Gewasklemsystemen [27], and consists of two parts; the Qlipr itself, and a metal wire with flared endstop. The Qlipr is a folded metal device with a foam insert, by its design the plant can be clamped with the foam and hung from the metal wire. Because of the endstop at the bottom of the metal wire the Qliprs can not slide off the metal wire. Two Qliprs are required per plant, as to support the plant when it is lowered. Although this system costs more and is less popular than the string system, it does offer a different method of supporting plants that, at first glance, seems to be more straightforward to automate. See figure 5b for an example of a tomato plant supported with the Qlipr system. For this system to be automated, as with the string system, an understanding of the plant's structure is necessary, but in contrast, branches are less important to be detected.


(a) **String method**: String is wrapped around the plant to support its weight as it grows vertically. String is stored and hung from a metal hook at the top of the grenhouse.

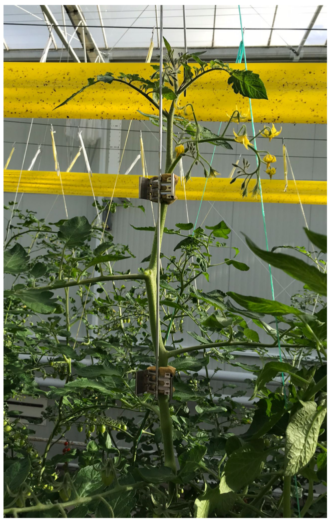

(b) **Qlipr method**: Plant is supported by two metal clips with foam on the inside, which are clamped around a metal wire with endstop for support.

Fig. 5: Two competing methods for supporting the plants weight for vertical growth. Where the string method is used most often, however the Qlipr method shows great potential for being automated.

Next to supporting the plants, the removal of: suckers and, low-hanging leaves is another task that shows potential to be automated. Suckers, which always start at the intersection of the stem and branch, are tiny tomato plant offshoots. If given enough time to develop, these offshoots can eventually grow into fully-grown tomato plants. With their own: stem, branches, flowers, and even new suckers.

As a result, the plant develops into a web rather than a single stem with branches. This type of tomato plant growth is not preferred by growers because it makes the plants more difficult to control. Additionally, because more nutrients and energy are expended on the plant rather than the growth of tomatoes, this decreases the overall efficiency of the tomato plant. Due to the fact that suckers can start at any node —the intersection of a branch and a stem— of the plant as well as being very small, finding and removing them requires a lot of labour. However, since suckers are always found at a plant node, we could use the structure of the plant to locate them. As the robot could focus in on these sections of the plant, allowing for a higher rate of detection.

Lower-placed tomato plant leaves, like suckers, lessen the plant's efficiency. Leaves placed lower on the plant receive significantly less light than those placed higher up due to the vertical growth and density of a greenhouse. As a result, the lower leaves of the plant grow larger in order to receive more light; however, this consumes the plant's nutrients and energy. These nutrients and energy, however, could be used to produce tomatoes; as a result, growers remove all leaves of the plant below the level of fully grown tomatoes. The removal of the leaves is again done at the node of the plant, removing the entire branch with its leaves in a single cut. Locating this node and branch using the plant's structure could provide an effective way of automating the removal of these leaves.

3.2 Robot Design

At the outset of this thesis, Lely had already designed and constructed a robot for work in a greenhouse. The requirements for this were that the robot is around 4 meters tall in order to reposition plants at the top of the greenhouse. In addition, the robot must be able to remove suckers and low-hanging leaves by traversing the entire height of the plant. These requirements led to the construction of the robot around a central stabilized pole with robotic arms that can move up and down using a rack and pinion system. This is all mounted to a cart that can ride around the greenhouse using the heating pipes and paths already available in modern greenhouses. Because of this, it was decided to use 4 degrees of freedom robotic arms with a SCARA design, which can only move in a plane parallel to the ground. In order to speed up operations the proposed design of the robot will include multiple arms working in conjunction. However, in order to reduce the complexity of the system, only one robotic arm was used for this thesis. See figure 6 for an overview of the robot's design. On the robotic arm sensors and an end-effector are placed at the tip, while an Nvidia Orin AGX single-board-computer [28] is located near the pole. This provides each robotic arm with enough computational power to perform tasks independently, while requiring minimal communication with a hub computer located in the cart. The end effectors can rotate around an axis colinear to that of the stabilized pole, which results in a robot with 5 degrees of freedom. For each of the tasks –repositioning, sucker removal, and leaf removal– a custom end effector is created. In order to reach in between the plants a small end effector is required, because of this Lely preferred the vision system to be as small as possible. With an additional preference for a single camera system to reduce the robot's cost.

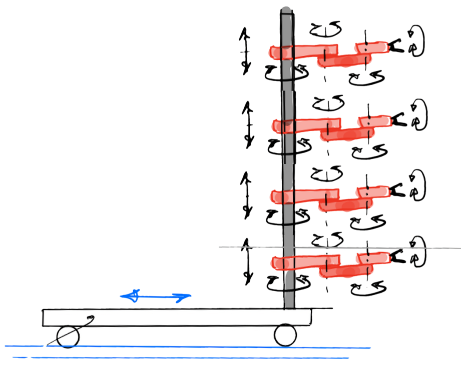


Fig. 6: Schematic overview of proposed greenhouse robot's design. Consisting of multiple SCARA robotic arms, which are able to move vertically on a stabilized pole attached to a motorized cart. The proposed design uses multiple arms to speed up operations. However, this thesis uses only a single robotic arm to reduce complexity of the system.

3.3 Data Acquisition

In order to start the process of generating plant structures input data for our system has to be gathered. The most apparent form of this input data is a point cloud recorded with a stereovision camera, as explained in the larger literature research, also found in appendix D. This method was utilized to collect data, and to do so, the robot arm was employed to log the stereo camera's position while capturing the point cloud. This was done to stitch the point cloud and to acquire a complete depth image of the plant. To choose the best stereovision camera, three commercial models were utilized and compared. The cameras compared were the Intel RealSense D435i [29], Stereolabs Zed 2 [30], and the Luxonis OAK-D [31]. But when the data was analyzed, an unexpected situation occurred; data from all cameras was insufficient. This is in contrast to what the earlier search for the best capture technique revealed. Out of the cameras tested the Luxonis OAK-D showed the best performance, however, all lacked in depth image detail. Most detail was unfortunately lost in the stems, which made them practically unusable for creating plant structures.
A smaller study was carried out to provide an answer to the following research question:

What is the cause for stereovision's low-depth sensing performance in a greenhouse environment?

To begin with, numerous stereovision algorithms exist, and it was unclear which one was employed by the cameras. As it could be that another algorithm could outperform it in a greenhouse environment. To see if another technique might perform better, the depth estimates were repeated using OpenCV algorithms. For this, images with both mono cameras on the OAK-D were captured, and a camera calibration was done with a standardized camera calibration checkerboard. For the depth estimation the following algorithms were tested: block matching, semi-global matching and feature matching. These algorithms were picked for

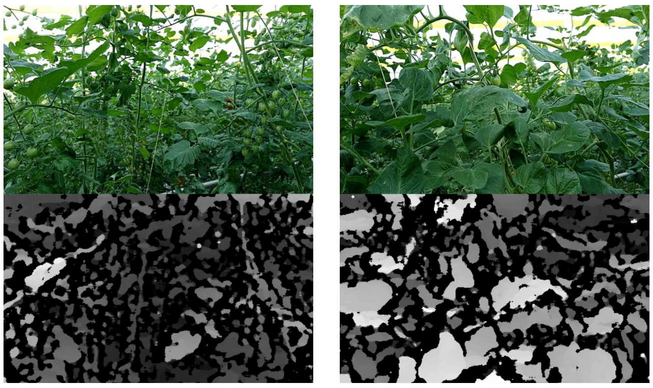


Fig. 7: Two examples of the Luxonis OAK-D's poor depth detection of stems, which performed the best of the cameras evaluated. Each example has the color camera at the top, and the bottom depicts the depth that has been detected; the lighter the pixel, the closer it is.

review as they come supported in the OpenCV library [32], which makes implementation fast and reliable.

Both block matching and semi-global matching use the camera's extrinsics in order to reduce the search field for the matchers. They both use a set of rectified images for matching, as here matches always lay on the same height in the image. Which allows for a small block of pixels to be compared between both the left and right image. Where block matching simply slides a block of pixels from the left image over the right image to find a match, semi-global matching also uses mutual information theory to improve matching between blocks [33]. Thus, in theory semi-global matching should result in better performance. Both speed and accuracy of the algorithms are greatly influenced by the size of the block. A smaller block enables for smaller details to be matched between images, but this implies that more comparisons must be conducted, slowing down the algorithm. The theory is that; because these calculations had to be made on the camera, the block size was constrained, making it impossible to match minor features on the stem between images. With the reimplementation we were able to test if changing the block size would lead to an improvement in the stems' depth estimation.

The outcomes of OpenCV's depth-calculating algorithms for block- and semi-global matching are shown in figure 8. When compared to the OAK-D camera's built-in depth estimation, an improvement can be seen here, as plant stems are clearly visible in the results. The number of frames per second at which this depth can be determined did suffer as a result, though. Because of this, all experiments above produce too few FPS to be used in combination with robotic control, with the possible exception of the 7.10 FPS for the block matching method with a block size of 5. However, this outlier does result in the lowest depth estimation performance across all tests, with particularly high levels of noise. In contrast to the hypothesis stated above –bigger blocks would result in higher FPS– the opposite is to be found true. Bigger blocks reduce the FPS of the algorithm. As the number of pixel comparisons is defined by the following equation:

$$c = B^2(I_w - B + 1)(I_h - B + 1) \tag{1}$$

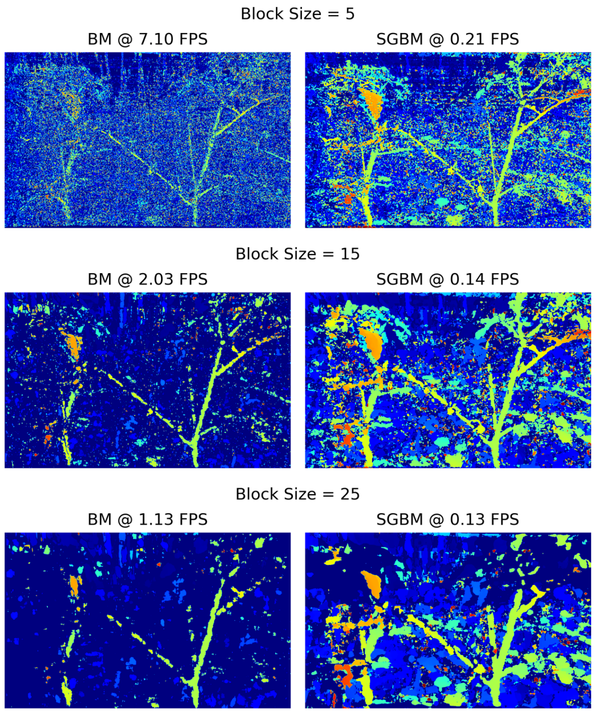


Fig. 8: Performance of disparity algorithms is compared. Larger block sizes perform better, but they can increase computation time and reduce resolution. Standard block matching on the left side, semi-global matching on the right.

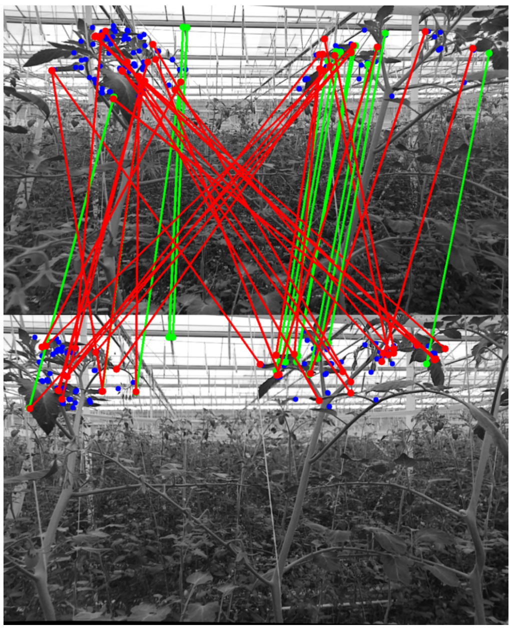


Fig. 9: Performance of ORB feature detection algorithm. For this example only 16 of the best 50 matches turn out to be correct. Correct matches given in green, incorrect in red, all keypoints show in blue.

With the number of comparisons given with c, block size B, image width, and height as I_w and I_h. Thus for our input images we get 6.296.400 and 144.760.000 comparisons for block sizes 5 and 25 respectively. An increase of approximately 23 times the amount of comparisons. Thus the reason that the OAK-D is not able to detect plant stems does not seem to be related to the block size that is used, but

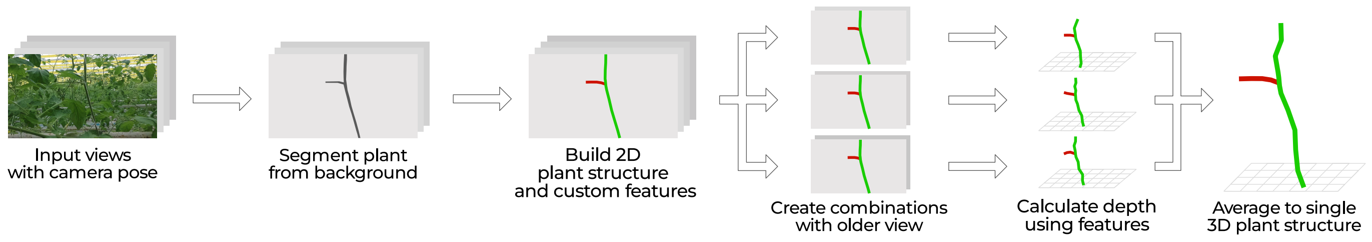


Fig. 10: Architecture of the algorithm used to create a 3D plant structure in a greenhouse environment.

rather has to be found in the implementation of the filtering algorithms used in the camera. As the code used on the OAK cameras is proprietary, no further investigation into the algorithms used can be done.

An alternative to matching image blocks is to match image features. This would result in a sparse field of depth data, but could provide enough information to build plant structures. There exist multiple ways of creating these features, the most common of which are ASIFT [34], SURF [35] and ORB [36] features. Literature shows that ORB outperforms ASIFT and SURF when it comes to both speed and correct matches [37]. This is also what a brief analysis of these feature detectors revealed, with ORB outperforming the previously mentioned detectors. An example of the performance of ORB feature detection and matching can be seen in figure 9. Even though the ORB feature detector had the greatest performance, it still leaves a lot of room for improvement when it comes to our input data. Only areas with strong contrast allowed for the detection of features; the image's lower portions, which are covered in vegetation, did not. As a result, the plant's stems nearly entirely lack features, which prevents us from gathering depth data here. In addition to this, the matcher has difficulty locating accurate matches even when features are found on both input photos. Other picture pairs exhibit the same behavior as the example in figure 9. Merely 12.53% of the matches in an experiment with 12 image input pairs are valid. Because of the poor performance, this method of creating the plant structure will not be investigated further.

4 Design

As previous attempts to collect depth data in the greenhouse did not produce satisfactory results, a new method must be developed for constructing plant structures. Using a robotic arm and a single camera, the objective is to develop an algorithm that can detect the position of a plant to within 20mm of its actual position. Lely specified 20mm of accuracy as a design requirement. As the proposed algorithm is composed of multiple interconnected components, a general overview of the entire design will be provided first. Each component will then be detailed in its own subsection.

4.1 Overview

The algorithm is built around creating custom features for the plant, these 2D features are then matched and combined to form the plant structure in 3D. For this, different views are needed from the plant, which is accomplished by moving the camera instead of using multiple cameras. For this, the robot's cameras move in a horizontal straight line parallel to the row of plants. Due to this movement and the assumption that the plant does not move, features can be matched based on the height of the feature in the image. Thus, a feature matcher can be omitted from the algorithm. The robot's design -multiple arms at multiple heights, all with their own camera- allows us to get the full overview of the plant in a single pass. Features are created from of a 2D structure –or skeleton– of the plant. This structure is created after segmenting the plant from the background with the use of a deep learning network. As multiple plants can be in the same view, a tracking algorithm is used to distinguish and match plants between views. Using these features and the pinhole camera model the 3D plant structure is created. Finally, multiple plant structure segments are combined into a single structure of the full plant. See figure 10 for a visualization of the algorithm's architecture, and how its components are used together.

4.2 Segmentation

As stated in the previous section, in order to create the 2D plant skeleton a segmentation -or outline- of the plant is used as an input. At the time of writing the best way to create such a segmentation is by using a deep learning network [38]. To select the best model for our greenhouse input data, several state-of-art models were trained and their performance compared. The following networks were used: Mask R-CNN a traditional convolutional neural network [39], DeeplabV3+ a transformer-based network [40], and finally PID-Net a neural network based on a PID controller's architecture [41]. The networks were trained on a custom dataset of about 600 fully annotated images. DeeplabV3+ performed the best of the three tested networks, with an accuracy of 77.2% and an intersection over union (IoU) of 83.2%. This model was supplied by Lely for use in this thesis. Inference of the model is implemented using Microsoft's ONNX Runtime Library [42]. ONNX Runtime enables neural networks built with various frameworks, such as Pytorch and Tensorflow, to be run and optimized on a wide range of hardware. Because of these optimizations and the GPU in the Jetson Orin, the DeeplabV3+ model can run at nearly 15 FPS. As with most segmentation networks, the output of the DeeplabV3+ model is in the form of an image. In order to go from a pixel-based segmentation to an outline OpenCV's `findContour()` function is used.

4.3 Plant Structure

After creating an outline for all of the plants in view, a 2D structure can be formed. The method for this was adapted from the work of Lewandowicz et al. [43], in this research a

centerline is created from the outline of rivers. Lewandowicz et al. proposes two methods either based on Voronoi diagrams [44] or Constrained Delauney Triangulation (CDT) [45]. For this thesis both methods were tested, in the end CDT was chosen as it created a more precise representation, especially when it comes to the thinner parts of the plant-like branches. The CDT works by splitting up the outline into triangles, triangles that all cross the centerline of the shape. By using these crossing edges a centerline can be generated by using the midpoints of these edges. The algorithm builds this centerline by first starting at the triangle with the lowest centroid and adding crossing edge midpoints of its neighboring triangle. This is done until either one of two types of triangles gets found, triangles with either 0 or 2 neighbors. As these triangles indicate the start or end of a branch/stem respectively and thus either ending or starting a segment of the plant structure. Note that by neighbors we exclude triangles that are already used in building the structure. By ”walking” every triangle of the CDT the complete 2D plant structure is created. See algorithm 1 for the pseudo code of the plant structure building algorithm, an example of a fully build plant structure can be found in figure 11 with all segments shown in a random color.
One disadvantage of utilizing the CDT is that it can produce a jagged centerline when there are a lot of points on the plant's outline, this is something that happens when converting the pixel-based segmentation to an outline. As every pixel creates a squared bump in the outline. Two methods of filtering are used to combat this, one on the outline and one on the final plant structure. The Douglas-Peucker algorithm is utilized for the outline, which decreases the points that lie on the outline while maintaining the general shape [46]. By lowering the number of points on the outline, we get both a smoother centerline and a reduction in the calculation required to create it, as fewer triangles must be validated. The algorithm works by creating a line between the first and last point of the outline, all points that lay inside of threshold distance perpendicular from this line are removed. The point furthest away from this line is then added, this time all points are checked for a threshold away from the two lines we just created. This is repeated until we end up with a simplified outline shape.

Nonetheless, using Douglas-Peucker can still result in a jagged 2D plant structure; as an illustration, look at the two points that are near to one another in the purple section of figure 11. These faults are created as a result of imperfections in the outline made by the neural network. To filter out these kinds of irregularities bezier smoothing is applied to all 2D plant structure segments. Here all points of the segments are used to create a bezier curve of order $n-1$ with n being the number of points in that segment, resulting us with a visually smooth 2D plant structure.

Algorithm 1: buildPlantStructure():

```
branches[], stem[];
developing_segments[] = lowestTriangle();
while developing_segments.length do
    for segement : developing_segments do
        if segment.neighbors == 0 then
            segment.add(outsideEdge().midpoint);
            branches.add(segment);
            developing_segments.remove(segment);
        else if segment.neighbors == 1 then
            segment.add(commonEdge().midpoint);
        else if segment.neighbors == 2 then
            for neighbor : segment.neighbors do
                developing_segments.add(
                 commonEdge().midpoint);
            stem.add(segment);
            developing_segments.remove(segment);
        else
            return error;
return branches, stem;
```

4.4 Depth Calculation

To transition from a two-dimensional to a three-dimensional plant structure, the algorithm makes use of computer vision triangulation. Note that this is different from the constrained Delauney triangulation we used in building the 2D plant structure. This type of triangulation –also known as reconstruction or intersection– refers to the process of determining a 3D point from its projection onto a set of images. A visualization of the function's workings can be found in figure 12.

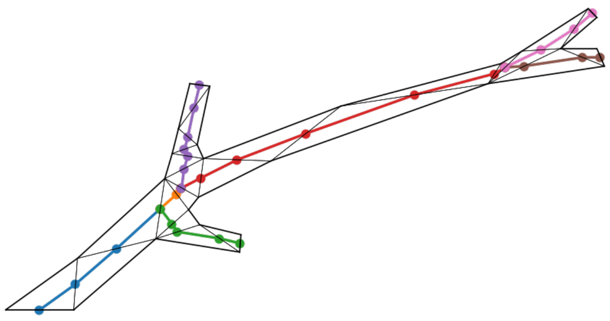
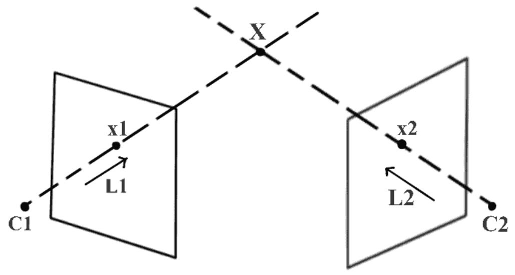


Fig. 11: Visualisation of plant structure build with our algorithm. Black lines show the constrained delauney triangulations and outline, with all plant structure segments shown in random colors. Note that the structure splits when the algorithm encounters a triangle without contour edges.



1

Fig. 12: Visualization of computer vision triangulation algorithm calculating depth at the intersection of lines passing through matched image points x1 and x2. Where X is the object and C1 and C2 are the respective camera apertures of both views. [47]

Triangulation works by projecting lines from the camera's aperture through an image point, this is done for all images in the set. The 3D point we want to calculate lies at the intersection of these lines, but only if all image points represent the same point in 3D space. See appendix A for a mathematical explanation of how the intersecting lines are constructed. In order for this equation to function correctly, no distortion can be present in the images. To combat this, the camera's distortion is measured for the camera and removed on the segmentation image using the rational model as described by Claus et al. [48]. Both the pinhole model and camera distortion were implemented using the OpenCV functions `triangulatePoints()` and `undistort()` respectively.

4.5 Feature Creation and Matching

As described in the section above in order for triangulation to correctly calculate a 3D point, the image pixel on which it is based, have to represent the same 3D point. In other words this means picking the same pixel on both input images corresponding to the same location on the same plant. We call these corresponding points features of the plant structure in this thesis. Hence, the issue of selecting corresponding features can be reduced to two challenges in the algorithm's design; plant tracking (to solve the same plant problem), and feature creation (to solve the same location on plant problem).

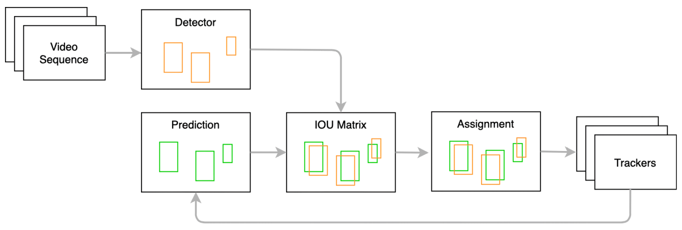


Fig. 13: SORT Tracking algorithm's architecture, implementing a kalman filter to track multiple bounding boxes over time. [49]

Tracking was solved using the Simple Online and Realtime Tracking algorithm, better known as SORT by Bewley et al. [50]. SORT was designed to track bounding boxes created by neural networks. It does this by using a hungarian algorithm with IoU as cost for assigning bounding boxes, and a Kalman filter to keep track of existing bounding boxes. The Kalman filter works by trying to predict the movement of the bounding boxes, it does this with the following state vector:

$$\mathbf{x} = \begin{bmatrix} u & v & s & r & \dot{u} & \dot{v} & \dot{s} \end{bmatrix}^T \tag{2}$$

Where u and v represent the bounding box' center position in its horizontal and vertical components. The states s and r represent the size (bounding box height times the width) and aspect ratio (bounding box height divided by width) respectively. The predictions work by taking the derivative of the bounding box's position $(\dot{u}, \dot{v})$, and size $(\dot{s})$. Multiplying this with a time step creates a prediction of where the bounding box is in the next frame, allowing for better tracking of the bounding boxes. As no derivative is given for the aspect ratio, this is expected to be constant between time steps. See figure 13 for an overview of the algorithm's architecture.

To go from our segmentation outline to a bounding box compatible with the SORT algorithm, a box is drawn around the extremes of the outline. Next, the bounding box's top and bottom are stretched to the top and bottom of the image, to decrease the variability in aspect ratio of the bounding box. With these adjustments, the SORT algorithm is able to consistently track multiple plants in the camera's view.

To solve feature matching between the same plant, we make use of the special movement that the camera makes in front of the plants. As described in section 4.1 during plant structure building the camera's movement is restricted to moving horizontally and parallel with the row of plants. This movement results in the camera seeing the same vertical section of the plant in all input photos. Therefore, the assumption is made that feature matching can be reduced to taking features at the same vertical height from the plant structure. In our algorithm, 24 features equally spaced on the image's vertical axis were interpolated from the plant structure's bezier curve. The 24 features were then used as an input for the `triangulatePoints()` function provided by OpenCV.

4.6 View Combining

The output of the system described above is likely to be noisy, as most notably the feature creation and camera location can generate noise in the system. Both of these errors are assumed to be gaussian distributed. As the error in both is a combination of multiple statistically independent factors. By taking the mean of multiple plant structure observations the system should converge to the actual location of the plant. In order to improve this means every new input is compared with multiple previously created 2D plant structures. Convergence should be faster and more robust with a bigger dataset, as described by the central limit theorem [51]. This is implemented by comparing every new input to at most 20 randomly picked previous observations of the same plant, we call this view combining. As an example; by having the camera move a distance of 1 meter in 5 seconds we end up with 1500 different 3D plant structures, which we then average to a single structure.

Because of the way triangulation works, the noise on the computed distance increases as the distance between the cameras decreases. The error induced can be calculated with the following equation:

$$\Delta Z = \frac{Z^2}{bf} \Delta D \tag{3}$$

Here Z is the distance to the object we are measuring, b and f are the baseline (distance between the cameras) and focal distance of the camera respectively. Finally, ΔD is the disparity error, or the amount of pixels we are incorrect in our feature matching. Figure 14 shows how the error grows when we decrease the baseline, for multiple object distances. Because of this rapid increase of the error for smaller baselines, we only combine views when the baseline is at least 100mm. Without it, we introduce a lot of noise into the system. This is also why, due to the baseline constraint, we can only obtain 20 random samples at most. Thus limiting our example above to 1350 3D plant structures, instead of the 1500 stated earlier.

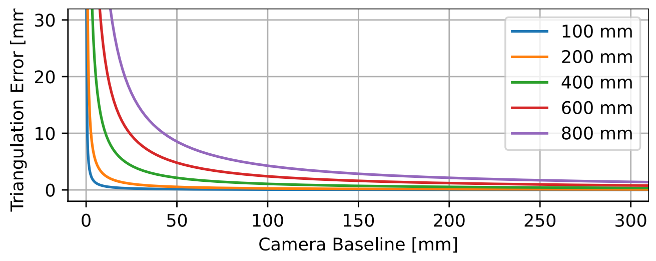


Fig. 14: Triangulation distance error for multiple object and baseline distances. Disparity error is constant with an error of 1 pixel.

4.7 Plant Combining

With the above-described algorithm, we obtain a number of smaller 3D plant structures. At least one from each robotic arm's, as every arm operates at a different height we end up with 3D plant structures from that height. Multiple segments can also form per robotic arm when plant parts are obscured by leaves or other plant parts. We'd like to combine these plant structure sections into a single large plant structure encompassing the whole plant. For this purpose, a metric was developed; when a pair of plant segments score below a certain threshold, their structures are merged. In this manner, occluded portions are filled in or the entire plant structure is lengthened using multiple camera views. The metric is run for every segment at the end of the camera's movement. Because of its shape, it is referred to as the flame metric, see also figure 15. The flame metric uses two vectors as input: $H_t\vec{L}_n$ between the nose (top) and tail (bottom) of two plant structure sections, and a normalized vector $\vec{L_n}$ projected out of the nose of the lower section. We then calculate the flame metric as follows:

$$\mathrm{FM} = |H_t\vec{L}_n| \cdot \left[1 + \arccos\left(\frac{\vec{L_n} \cdot H_t\vec{L}_n}{|\vec{L_n}||H_t\vec{L}_n|}\right)\right] \qquad (4)$$

This equation can be read as the distance between the nose and tail of a section multiplied by the angle between the two vectors with an additional distance. See figure 15 for examples of the metric's performance. Note that this is a 2D visualization, while the actual metric is calculated in 3D. Through trial and error the threshold for the flame metric was set at 750, resulting in full plant-length structures created with our algorithm.

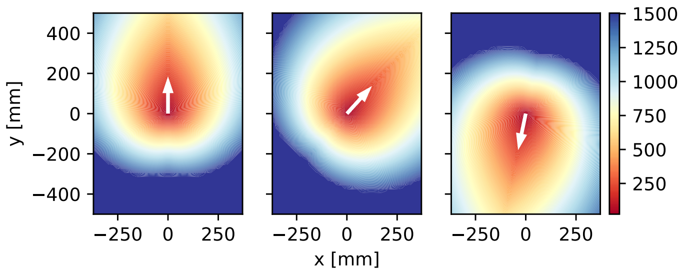


Fig. 15: 2D Visualisation of the flame metric output for three different $\vec{L_n}$ vectors. Metric value given in color, with $\vec{L_n}$ in white.

5 Method

Now that the algorithm is designed and implemented, its performance has to be evaluated. In the following section the testing method for the algorithm's dependent variables speed and accuracy will be explained. Followed by section 6 where the results of the experiments are given and elaborated upon.

5.1 Speed

As stated in section 4.3, the use of the Douglas-Peucker algorithm did not only result in less jagged plant structures but also sped up the algorithm. With Douglas-Peucker implemented the algorithm creates fewer triangles, resulting in a reduced search space while building the plant structure. The speed of the algorithm was measured in microseconds per plant structure generation, proportional to the size of the outline. A dataset of 1386 plant outlines was created from a recording in the greenhouse. For each of these outlines, the algorithm's calculation time was recorded, both with and without simplifying the outline.

5.2 Accuracy

The proposed robot will use a 3D plant structure to navigate and locate plant features. Other plant structure generation algorithms, as described in section 2, did not provide data on the output's accuracy. A high level of accuracy is crucial to the functionality of the robot, as poor accuracy would result in the robot improperly positioning the end-effector. In order to demonstrate the efficacy of our algorithm, it is essential to measure its precision. Experiments to examine the accuracy of our 2D and 3D plant structures are described in the following sections.

5.2.1 Two-Dimensional

Due to the fact that the 3D plant structure is composed of numerous 2D plant structures, it is preferable that the latter be as precise as possible. For this taking the exact center of the plant's outline would result in repeatable features. Without repeatable features, feature matching between images will not work as expected, preventing triangulation from calculating precise depths. Figure 16 demonstrates a few examples of 2D plant structures. Upon visual inspection, the 2D algorithm generates the correct plant structures.
A study was conducted to evaluate the accuracy of the 2D plant structure algorithm in order to verify our assertions. For this purpose, a dataset of 482 imitation plants was compiled, and for each plant outline, three structures were generated: one with the raw output of the neural network as the outline, one with Douglas-Peucker simplification applied to the outline, and one with Bezier-Curve filtering applied to the output. Imitation plants were used in the creation of this dataset to negate the variability of the neural network's output. Our structure's centrality in the original outline served as a metric for measuring precision. The utilized metric is $M_{\text{center}} = \frac{D}{W}$. Here, W represents the horizontal distance between the two sides of the outline, and D represents the horizontal distance from the centerline to one of the outlines. Thus, the output of the metric is 0.5 when the centerline is precisely in the center, and 0

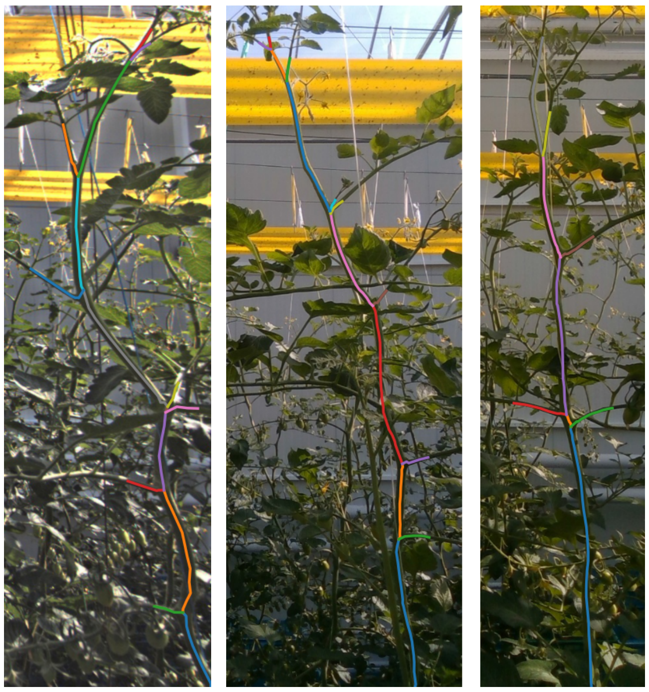


Fig. 16: Visualisation of 3 plant structure generated by the 2D algorithm. Plant structure segments are randomly colored to indicate splitting of the structure.

or 1 at the plant's edges. M_{center} was then calculated for 24 measurement locations evenly spaced across the entire height of the image. As there is more data available to the algorithm, it is anticipated that the original outline will outperform the other two methods. It must be determined if the accuracy loss caused by simplifying the outline outweighs the performance increase.

5.2.2 Three-Dimensional

The developed algorithm was designed to precisely detect the plant's position in 3D space. To confirm this, two tests measuring the accuracy of our algorithm were conducted. For the test setup, a stereovision camera was used to establish the ground truth for both an imitation plant and a rigid plant structure. Utilizing a large television as a backlight allowed for accurate segmentation based on the image's brightness. To improve the stereovision camera's depth-sensing performance, an image of Gaussian blur was used to provide texture. Using the brightness-based segmentation as a mask, the stereovision point cloud was sampled. Resulting in a point cloud formed around our imitation plant. This is converted to a ground truth by sampling the mean of this point cloud at 24 distinct heights, the same as the number of features we create with the algorithm.

As stated previously, two custom plant varieties were utilized for the experiments. First, an imitation plant made from flexible pneumatic tubing was suspended in front of the television. This was used to visualize the algorithm's performance on various plant structure configurations. Four distinct configurations of this artificial plant, each with distinct curvatures, were recorded and analyzed.

Secondly, a rigid plant structure comprised of straight, welded metal rods was utilized. Prior to assembly, the rods were cut at a 45-degree angle to create a spiral-shaped

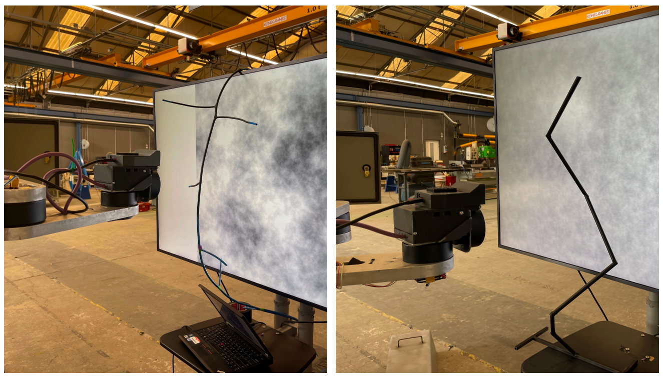


Fig. 17: Experiment setup for both imitation and rigid plant structure accuracy measurements

structure. The purpose of the fixed plant structure was to test the accuracy as a function of camera-to-plant distance. As a result, this information could be used to determine how many cameras and, consequently, arms the robot requires to cover the entire plant. Two recordings at three distances from the center of the rigid plant structure were produced for this purpose. The distances are labeled Close (200mm), Medium (400mm), and Far (600mm). These distances were selected because they reflect the actual distances the plant can be located in the greenhouse. Figure 17 shows the experimental setup for both imitation and rigid plant structures.

For both the rigid, and imitation plant recordings were made passing the camera in a straight line in front of the plant. For every matched feature, the algorithm creates its own point in 3D space at which it detects the plant. We can measure the accuracy by calculating the error for every point with respect to the ground truth. This was done by calculating both the X and Y distance to the ground truth in a horizontal plane at the point's Z height. Due to the fact that the ground truth consists of only 24 specific points, we interpolate between these points to calculate the error for each calculated point. For all errors the Mean, Median, Standard Deviation (SD) and RMSE were calculated.

5.3 Camera positional error

If the results indicate that our 2D plant structure construction algorithm can correctly identify the plant's center, then large errors in depth can only be attributed to an incorrect camera location. This can occur when the arm's actuators and sensors return incorrect angle values, or when flex occurs in the robot as a whole. Flex has a possibility of developing in either the arm or more likely the pole it is affixed to. The segmented point cloud of the plant is utilized once more to demonstrate that errors are in fact caused by this positional error. As this point cloud was generated from a stationary object, each point should be positioned at the ground truth. Any deviation from this would indicate that the camera's position is inaccurate. By calculating the standard deviation between our ground truth and each point in the point cloud, we can estimate the camera's positional error. The Pearson correlation coefficient will be utilized to analyze the relationship between the positional error of the camera and the algorithm.

6 RESULTS

This results section unveils the findings of this thesis, presenting a concise analysis of the collected data. It highlights both the 2D and 3D plant structure detection performance. In addition to the speed enhancements made.

6.1 Speed

In figure 18, the calculation times for a dataset of 1386 plant structures, taken from real-life recordings in the greenhouse, are represented graphically. Fitting the data points on the graph reveals that the calculation time complexity for both tests is $\mathcal{O}(n)$. Note that this is not the actual algorithm's complexity, but rather its calculation time; this is merely an approximation of the algorithm's complexity.

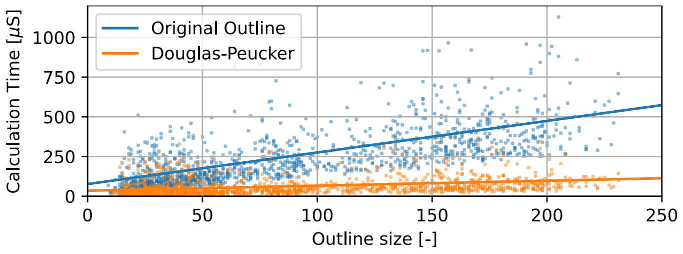


Fig. 18: Plot of calculation time and fit-line for both the original plant outline and the Douglas-Peucker simplified outline. By implementing simplification calculation time is reduced by a factor 3.965. Both scale with $\mathcal{O}(n)$, with n the size of the outline.

Even though the complexity of both versions of the algorithm is identical, the plot reveals a significant difference in calculation time. To evaluate this mean, the standard deviation was calculated for both datasets. The implementation of Douglas-Peucker reduces the average calculation time by 396.5%, from 246.63 to 62.197 μs. The standard deviation decreases from 364.27 to 90.643 μs, a decrease of 401.9%.

6.2 Two-Dimensional Accuracy

The 482 plant outlines were used to collect 5033 centerline measurement points. For this, the imitation plant was used to create the dataset. For each of these measurements, we calculated the centrality metric M_{center}. The outcomes of this investigation are displayed in table 1, and figure 19.

TABLE 1: Comparison of centrality metric's M_{center} mean and standard deviation for three different methods of improving the performance of the plant structure algorithm. Both metrics given in pixels.

	Original	Simplified	Bezier
Mean	0.4942	0.4976	0.4813
Standard Deviation	0.0601	0.0885	0.1438

In contrast to expectations, the simplified outline outperforms the original outline in terms of precision, with a mean that is closer to 0.50. This can be attributed to the increased complexity of the original outline, as minor "bumps" in this metric can cause a decrease. However, the original outline outperforms the other two methods in terms of precision. When the centerline is Bezier-smoothed, there is a significant loss in measurement performance, both in terms of accuracy and precision. Contrary to what a visual

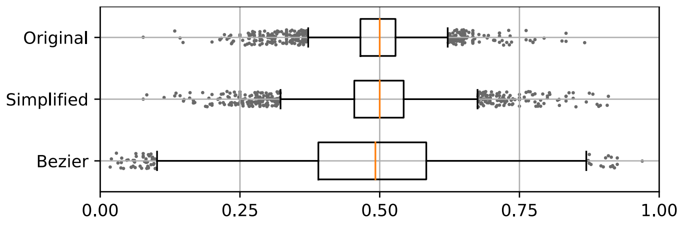


Fig. 19: Boxplot of 2D accuracy metric for 3 different methods: Using the original outline, Simplification of the outline using Douglas-Peucker, and Bezier smoothing on the generated centerline.

inspection of the plant structures would suggest. In light of these findings, Douglas-Peucker outline simplification will be incorporated into the final design, as the speed increase –explained in the prior section– outweighs the minor loss in measurement performance. Bezier-smoothing will not be implemented in the final design, as it does not improve the algorithm's measurement performance.

6.3 Three-Dimensional Accuracy

As described in section 5.2.2, two man-made plant structures where tested. A flexible plant structure made out of pneumatic tubing being called the imitation plant in this thesis. Secondly, a rigid plant structure to analyze the algorithms performance as a function of distance was used. For all tables and figures, the world frame is defined as follows: Z is the vertical axis, X is the horizontal axis as seen from the camera, and the Y-axis is the depth as seen from the camera, with the frame's origin at the base center of the robot's pole.

6.3.1 Imitation Plant

For the imitation plant, four distinct recordings were produced, each time altering the shape. Table 2 displays statistics for our recordings. We observe that there is no relationship between the number of recorded frames and the generated point. Several factors can explain this, including segmentation performance, inter-frame recording distance, and tracking performance.

TABLE 2: Comparison of measured points for every imitation plant recording

	Plant 1	Plant 2	Plant 3	Plant 4
Frames	103	113	112	104
Points	7735	12490	9501	3915

As a preliminary outcome, we will examine the output of one of the recordings, for which imitation plant 2 was utilized. See figure 20 for a front and side view of the ground truth in orange and all measurements from our algorithm in gray; section B.1 contains graphs for all other recordings.
The variability of depth is significantly greater than that of width, as depicted in the figure. To support this hypothesis, we have measured the error for every point in the XY-plane relative to the ground truth. This study's results for all imitation plants are available in table 3 and figure 21. These support our earlier assumptions, most notably that the standard deviation in Y is greater than in X. Across all plants, the standard deviation along the Y axis is roughly twice as

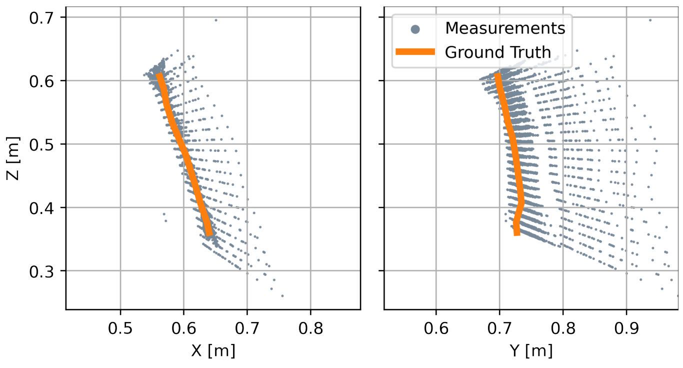


Fig. 20: Front (XZ) and side (YZ) view of the ground truth and measurements from the same imitation plant.

large. The Pearson r-value for the correlation between the standard deviations of X and Y is 0.9902 with a p-value of 0.0098, indicating a positive correlation between errors in X and the depth estimation in Y. This illustrates that small errors in either camera position or feature matching are enlarged in the depth estimation.

With the exception of recording number 4, our mean and median remain close to the actual values for the majority of the recordings. Here, the mean and median values are significantly greater than in all other recordings, with the majority of deviations in the Y direction.

TABLE 3: Accuracy for errors to the ground truth in imitation plant structures, all measurements given in mm.

	Median		Mean		SD		RMSE
	X	Y	X	Y	X	Y	
Plant 1	-1.410	6.911	4.071	16.858	34.88	59.77	71.34
Plant 2	0.656	7.798	1.139	11.79	6.502	23.15	26.81
Plant 3	1.587	7.406	0.643	11.92	8.368	25.05	28.98
Plant 4	12.46	-20.17	22.99	-41.24	27.39	55.96	78.17
All	0.778	6.894	3.066	9.692	20.13	40.44	46.30

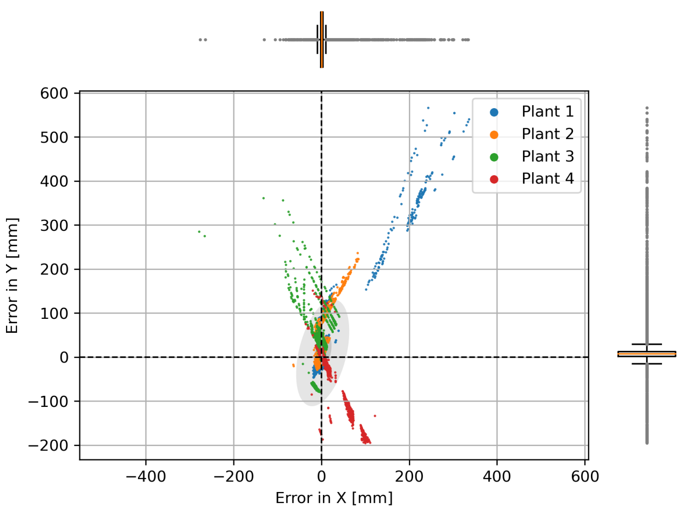


Fig. 21: Plot of errors to the ground truth for all fake plants.

6.3.2 *Accuracy with respect to distance*

As was the case with the artificial plant structures, multiple recordings were made of the fixed plant structures. Two separate recordings were made for each distance (200, 400, and 600mm) with the following names: Close, Medium, and Far, followed by a number for each cycle. Statistics from the recordings can be found in table 4.

TABLE 4: Comparison of measured points for every fixed plant structure recording.

	Close		Medium		Far	
	1	2	1	2	1	2
Frames	175	174	198	172	177	157
Points	611	5840	179571	71301	142177	189680

We identify a correlation between the recording distance and the number of measurement sites for the artificial plant structures. This is due to the inter-frame recording distance; as we move closer to the plant, a shorter horizontal distance is captured. When this threshold is exceeded, the formation of points grows rapidly. As with the artificial plant structures, the measurement errors between our algorithm and a real plant were investigated. This study's findings are presented in table 5 and figure 22. Figures for all recordings can be found in section C.2.

TABLE 5: Accuracy for errors to the ground truth in fixed plant structures, all measurements given in mm.

	Median		Mean		SD		RMSE
	X	Y	X	Y	X	Y	
Close	5.566	24.37	3.919	30.73	10.25	31.09	45.08
Medium	0.984	15.55	-0.066	19.49	10.36	35.75	42.03
Far	0.189	-21.81	-0.808	-19.63	20.84	77.54	82.66
All	0.716	7.424	-0.452	-2.787	17.17	66.07	68.33

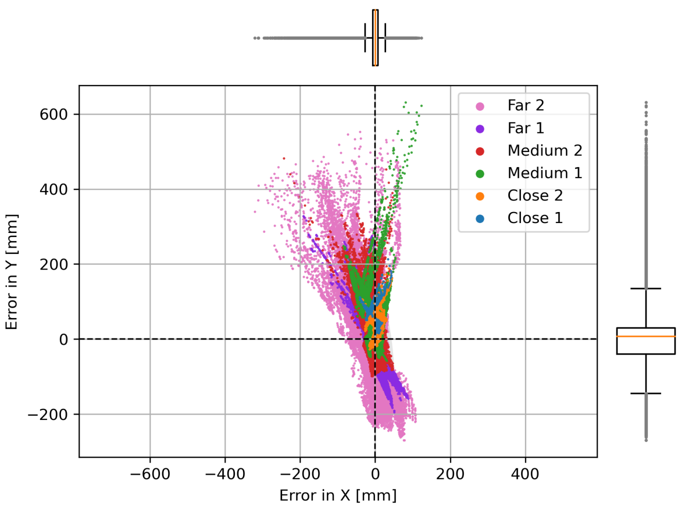


Fig. 22: Plot of errors to the ground truth for all fake plants.

Again, these results demonstrate that Y has a larger standard deviation than X. However, this time by a factor of 3.848, which is nearly double that of the artificial vegetation. With a Pearson correlation r-value of 0.9671, with a p-value of 0.00161, these two metrics continue to exhibit a strong correlation. Again demonstrating that imperfections in camera location or feature matching are exaggerated in depth calculation performance. We also observe another characteristic of the standard deviation, namely, the rapid expansion of both metrics in X and Y for far recordings. Unlike the measurements for the imitation plant, there appears to be no outlier when examining the mean and median values.

6.4 Camera positional error

For each recording, the error between the ground truth and the point cloud was calculated. As an illustration, figure 23 displays the ground truth as well as the point cloud from which it is derived for the recording Close 1. This example was selected because it demonstrates that a positional error occurred during recording, as the point cloud displays three distinct clusters of the same plant in X. We also observe that there does not appear to be the same clustering in Y, suggesting that the positional error only occurs in the X-direction.

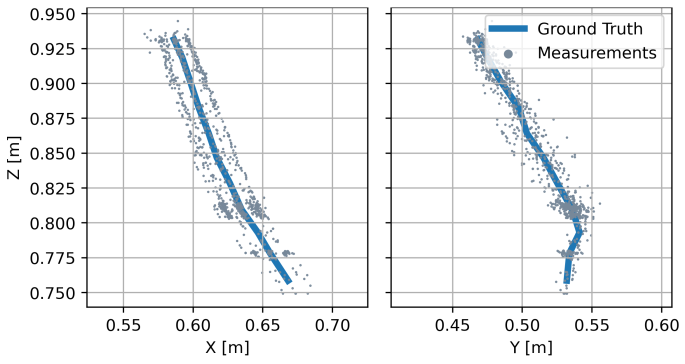


Fig. 23: Front and side views of the segmentated point cloud which is used to create the ground truth for recording Close 1. Note the positional error in X, as 3 clusters of the same plant structure can be seen.

To demonstrate a correlation between our measurement error and the positional error of the camera, the Pearson correlation coefficient is between the standard deviation of the ground truth's point cloud and the resulting algorithms output. With a r-value of 0.7649 and a p-value of 0.00994 correlation between these two metrics is proven. Thus indicating that the error in Y is a result of the camera's positional error in X.

7 Discussion

This thesis research has provided insight into the performance of an algorithm that detects plant structures using a single camera and robotic arm. This section presents a reflection on the research process.

The results indicate that we can precisely create a 2D plant structure. The 2D algorithm is shown to detect the centerline of a plant's outline quickly and precisely. By utilizing Douglas-Peucker outline simplification, the average calculation time is reduced by 401.9% to 62.197 μs, or approximately 16078 plant structures per second. The algorithm detects the centerline with an average deviation of less than 1% from the actual center. In addition, it can differentiate between a stem and a branch of the plant, which can be used to narrow the search space for the plant's suckers.

However, the results for 3D plant detection are less favorable. Even though the median and mean of our detection approach the ground truth when taken across all recordings, the same cannot be said for individual recordings. Where, a maximal mean offset to the ground truth of 67.68 mm is identified, with a maximum standard deviation of 83.17 mm. In addition to the fact that not a single RMSE value is less then the previously established maximum error of 20 mm, it is evident that the required precision for use with a robot is not met.

The resulting error is shown to be predominantly the result of improper camera location. Inaccurate actuator measurements or flex in the robotic arm and central pole can cause errors. In addition to the fact that all measurements for this thesis were taken with a single robotic arm, mounted on a 2-meter-high central pole, it is reasonable to assume that performance in a greenhouse would be worse. In this scenario, the pole height is increased to 4 meters and multiple moving arms are affixed to it, both of which would cause the camera's position to be less accurate.

Inaccurate plant feature matching is also adverse to depth sensing performance. As matches are based exclusively on height in the image, errors here could induce errors in our depth estimation. These do not, however, explain the significant errors we discover in our recordings. Using the recording Medium 1 as an example, the greatest error in the Y direction exceeds 600 mm. Using equation (3), we obtain a maximum error of 1.067 mm per pixel for a 100 mm baseline. Thus resulting in a match made by our algorithm that exceeds 562 pixels, or an error greater than the height of the image. Therefore, it seems highly unlikely that the depth estimation errors are solely caused by inaccurate feature matching.

By mounting multiple cameras on the same robotic arm, future research could eradicate both estimation errors. By rigidly connecting two or more cameras to one another, the positional error between the cameras is eliminated. Thus, significantly decreasing the substantial depth estimation error. With the implementation of image rectification between the cameras, feature matching based on height could persist in the algorithm. As with this method of image reprojection, corresponding points have identical vertical coordinates in an image.

A disadvantage of using multiple cameras is that all output images must be processed by the segmentation network. As the neural network is currently the system's bottleneck, a solution for the network's speed must be identified. With the current implementation, the detection rate of our algorithm for a two-camera system would be reduced to approximately 7 frames per second. Batching images in the network could provide a starting point for this research.

Additionally, future research could concentrate on detecting the structure of branches in three dimensions, since the present implementation of the algorithm only creates a structure for the plant's stem. Given that most of the branches in the image are horizontal rather than vertical, a new method of matching these features must be developed.

8 Conclusion

With the global demand for food increasing and labour costs rising, it is desired to automate this process. With delicate vegetation found in greenhouses, getting an accurate understanding of plant spatial structure is vital to creating a solution. This thesis proposes a solution to this plant structure detection problem. The solution is based on a robotic arm with a single camera attached. By use of a neural network to segment plants from the background, custom features were made on the plant's stem. Features were created based on the centerline of the plant's outline. By evenly spreading these features across the height of the image, an approximate match could be made. This was made possible by only allowing the camera to move in a straight horizontal path in front of the row of plants. The custom features allow us to use triangulation to detect the depth of the plant's structure.

Analyzing the output of both our 2D and 3D plant structures to a ground truth gave an insight into the performance of the designed algorithm. These results show that the algorithm is able to approximate the plant's structure. However, the variability in our measurements is too large, and thus the requirement of an accuracy of 20 mm from the ground truth is not met. Large errors in depth detection occur due to inaccurate camera location induced by inaccurate actuator position or flex in the robot's component. Future research could elaborate on these findings by studying the use of multiple cameras on a single arm. This could provide a way of eliminating the positional error between the camera views and thus improve accuracy.

References

[1] H. R. Max Roser and E. Ortiz-Ospina, "World population growth," *Our World in Data*, 2013. [Online]. Available: https://ourworldindata.org/world-population-growth

[2] G. D. Spencer, L. J. Krutz, L. L. Falconer, W. B. Henry, C. G. Henry, E. J. Larson, H. C. Pringle III, C. J. Bryant, and R. L. Atwill, "Irrigation Water Management Technologies for Furrow-Irrigated Corn that Decrease Water Use and Improve Yield and On-Farm Profitability," *Crop, Forage & Turfgrass Management*, vol. 5, no. 1, p. 180100, 2019.

[3] S. Sanford, "Reducing greenhouse energy consumption— An overview."

[4] "USDA ERS - Farming and Farm Income," https://www.ers.usda.gov/data-products/ag-and-food-statistics-charting-the-essentials/farming-and-farm-income/.

[5] M. Qaim and D. Zilberman, "Yield Effects of Genetically Modified Crops in Developing Countries," *Science*, vol. 299, no. 5608, pp. 900–902, Feb. 2003.

[6] N. O'Connor and K. Mehta, "Modes of Greenhouse Water Savings," *Procedia Engineering*, vol. 159, pp. 259–266, 2016.

[7] V. Maiwald, V. Vrakking, P. Zabel, D. Schubert, R. Waclavicek, M. Dorn, L. Fiore, B. Imhof, T. Rousek, V. Rossetti, and C. Zeidler, "From ice to space: A greenhouse design for Moon or Mars based on a prototype deployed in Antarctica," *CEAS Space Journal*, vol. 13, no. 1, pp. 17–37, Jan. 2021.

[8] R. Hevko, I. Tkachenko, S. Synii, and I. Flonts, "Development of design and investigation of operation processes of small-sclale root crop and potato harvesters," vol. 49, pp. 53–60, Jan. 2016.

[9] C. Zhi, H. FuPing, W. FengDe, S. WenFeng, and C. JunWei, "Development of technology and equipment of corn harvester in China." *Nongye Jixie Xuebao = Transactions of the Chinese Society for Agricultural Machinery*, vol. 43, no. 12, pp. 44–50, 2012.

[10] Q. O. Ogunlowo and J. O. Olaoye, "Development and performance evaluation of a guided horizontal conveyor rice harvester," *Agrosearch*, vol. 17, no. 1, pp. 66–88, Jul. 2017.

[11] A. Fujisawa, Y. Chida, Y. Nakamura, K. Hirano, T. Tsuchiya, T. Hatakeyama, T. Yamaguchi, K. Iizuka, T. Shimada, and S. Kitazawa, "Motion analysis of the root-cutting blade for an automatic spinach harvester: 12th International Conference on Motion and Vibration Control, MOVIC 2014," *MOVIC 2014 - 12th International Conference on Motion and Vibration Control*, 2014.

[12] Roger Chagnon, P. Eng., Dr Marie Thérèse Charles, Sylvain Fortin, P. Eng., Jérôme Boutin, P. Eng., and Isabel Lemay And Dominique Roussel, "Development of a Cabbage Harvester," in *2004, Ottawa, Canada August 1 - 4, 2004*. American Society of Agricultural and Biological Engineers, 2004.

[13] T. Hein, "Grow pipes - an energy advantage," https://www.greenhousecanada.com/grow-pipes-the-energy-advantage-20303/, Dec. 2012.

[14] S. Snow, A. Boyson, M. Felipe-King, O. Malik, L. Coutts, C. J. Noakes, H. Gough, J. Barlow, and M. C. Schraefel, "Using EEG to characterise drowsiness during short duration exposure to elevated indoor Carbon Dioxide concentrations," p. 483750, Nov. 2018.

[15] B. A. Eveleens-Clark, J. A. Dieleman, A. de Gelder, A. Elings, J. Janse, T. Qian, P. Lagas, and J. W. Steenhuizen, "Effecten van verneveling op groei en ontwikkeling van tomaat : teelt van eind april tot eind augustus," 2009.

[16] B. Arad, J. Balendonck, R. Barth, O. Ben-Shahar, Y. Edan, T. Hellström, J. Hemming, P. Kurtser, O. Ringdahl, T. Tielen, and B. van Tuijl, "Development of a sweet pepper harvesting robot," *Journal of Field Robotics*, vol. 37, no. 6, pp. 1027–1039, 2020.

[17] J. Jun, J. Kim, J. Seol, J. Kim, and H. I. Son, "Towards an Efficient Tomato Harvesting Robot: 3D Perception, Manipulation, and End-Effector," *IEEE Access*, vol. 9, pp. 17 631–17 640, 2021.

[18] "Greenhouse Vegetable Production — New Mexico State University - BE BOLD. Shape the Future." https://pubs.nmsu.edu/_circulars/CR556/index.html.

[19] "Development of the End-Effector of a Picking Robot for Greenhouse-Grown Tomatoes," *Applied Engineering in Agriculture*, pp. 1001–1009, Dec. 2013.

[20] J. Gao, F. Zhang, J. Zhang, T. Yuan, J. Yin, H. Guo, and C. Yang, "Development and evaluation of a pneumatic finger-like end-effector for cherry tomato harvesting robot in greenhouse," *Computers and Electronics in Agriculture*, vol. 197, p. 106879, Jun. 2022.

[21] P. Sammons, T. Furukawa, and A. Bulgin, "Autonomous Pesticide Spraying Robot for use in a Greenhouse," Oct. 2005.

[22] B. Neubert, T. Franken, and O. Deussen, "Approximate image-based tree-modeling using particle flows," in *ACM SIGGRAPH 2007 Papers*, ser. SIGGRAPH '07. New York, NY, USA: Association for Computing Machinery, Jul. 2007, pp. 88–es.

[23] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan, "Image-based tree modeling," in *ACM SIGGRAPH 2007 Papers*, ser. SIGGRAPH '07. New York, NY, USA: Association for Computing Machinery, Jul. 2007, pp. 87–es.

[24] J. Guo, S. Xu, D.-M. Yan, Z. Cheng, M. Jaeger, and X. Zhang, "Realistic Procedural Plant Modeling from Multiple View Images," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 2, pp. 1372–1384, Feb. 2020.

[25] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang, "Image-based plant modeling," in *ACM SIGGRAPH 2006 Papers*, ser. SIGGRAPH '06. New York, NY, USA: Association for Computing Machinery, Jul. 2006, pp. 599–604.

[26] D. Dey, L. Mummert, and R. Sukthankar, "Classification of plant structures from uncalibrated image sequences," in *2012 IEEE Workshop on the Applications of Computer Vision (WACV)*, Jan. 2012, pp. 329–336.

[27] Pellikaanq.com, "Qlipr: the ideal metal tomato clip," https://www.pellikaanq.com/products/qlipr.

[28] "NVIDIA Jetson Orin," https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/.

[29] "Depth Camera D435i – Intel® RealSense™ Depth and Tracking Cameras," https://www.intelrealsense.com/depth-camera-d435i/.

[30] "ZED 2 - AI Stereo Camera," https://www.stereolabs.com/zed-2/.

[31] "OAK-D," https://shop.luxonis.com/products/oak-d.

[32] "OpenCV," https://opencv.org/.

[33] H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[34] G. Yu and J.-M. Morel, "ASIFT: An Algorithm for Fully Affine Invariant Comparison," *Image Processing On Line*, vol. 1, pp. 11–38, Feb. 2011.

[35] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *Computer Vision – ECCV 2006*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer, 2006, pp. 404–417.

[36] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*. Barcelona, Spain: IEEE, Nov. 2011, pp. 2564–2571.

[37] E. Karami, S. Prasad, and M. Shehata, "Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images," Oct. 2017.

[38] U. Sehar and M. L. Naseem, "How deep learning is empowering semantic segmentation," *Multimedia Tools and Applications*, vol. 81, no. 21, pp. 30 519–30 544, Sep. 2022.

[39] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," Jan. 2018.

[40] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," Aug. 2018.

[41] J. Xu, Z. Xiong, and S. P. Bhattacharyya, "PIDNet: A Real-time Semantic Segmentation Network Inspired from PID Controller," Jun. 2022.

[42] "ONNX Runtime," https://onnxruntime.ai/docs/.

[43] E. Lewandowicz and P. Flisek, "A Method for Generating the Centerline of an Elongated Polygon on the Example of a Watercourse," *ISPRS International Journal of Geo-Information*, vol. 9, no. 5, p. 304, May 2020.

[44] W. Pokojski and P. Pokojska, "Voronoi diagrams – inventor, method, applications," *Polish Cartographical Review*, vol. 50, pp. 141–150, Sep. 2018.

[45] L. Paul Chew, "Constrained delaunay triangulations," *Algorithmica*, vol. 4, no. 1, pp. 97–108, Jun. 1989.

[46] M. Visvalingam and J. D. Whyatt, "The Douglas-Peucker Algorithm for Line Simplification: Re-evaluation through Visualization," *Computer Graphics Forum*, vol. 9, no. 3, pp. 213–225, Sep. 1990.

[47] "Introduction to Epipolar Geometry and Stereo Vision — LearnOpenCV #," Dec. 2020.

[48] D. Claus and A. Fitzgibbon, "A Rational Function Lens Distortion Model for General Cameras," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. San Diego, CA, USA: IEEE, 2005, pp. 213–219.

[49] Hyun-je, "SORT-ros," Dec. 2022.

[50] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sep. 2016, pp. 3464–3468.

[51] S. G. Kwak and J. H. Kim, "Central limit theorem: The cornerstone of modern statistics," *Korean Journal of Anesthesiology*, vol. 70, no. 2, pp. 144–156, Apr. 2017.

Appendix A
Pinhole Camera Model

The pinhole camera model allows us to create projection lines by using the camera's intrinsic en extrinsic parameters. It can be written as follows:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5}$$

The equation is comprised of 2 homogenous coordinates, a camera matrix and a rotation-translation matrix. Where the 2D homogenous coordinate represents the image point (x, y), with the 3D coordinate representing the line passing both through the camera aperture and (X, Y, Z). The rotation-translation or extrinsics matrix represents the camera pose as it relates to a world frame, consequently the line intersection is also in the world frame. The camera or intrinsics matrix is a 3-by-3 matrix comprising of c_x, c_x, f_x and f_y. Where c is the principal point –usually the image center– and f is the focal distance, both given in their x and y components. The line passing through the camera aperture and (X, Y, Z) can then be used for intersection when used for computer vision triangulation.

Appendix B
Imitation Plant Structure

B.1 Front and Side Views

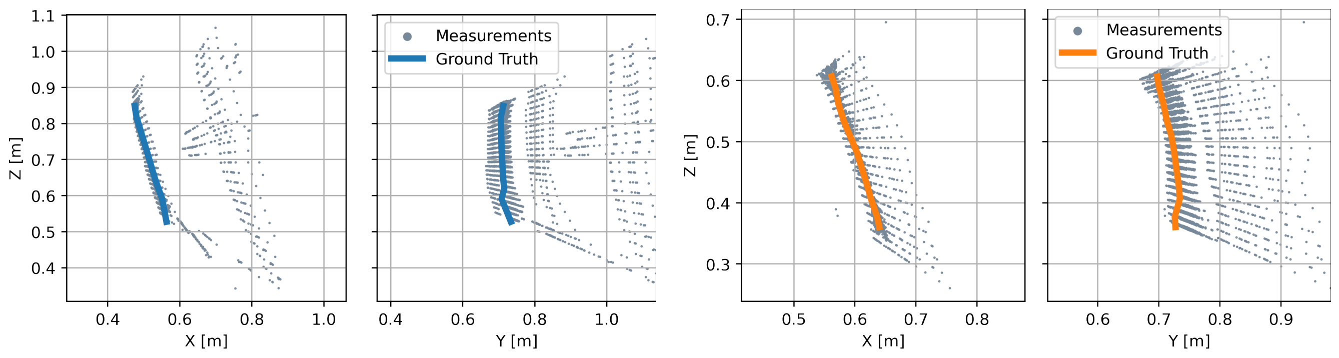


Fig. 24: Front (XZ) and side (YZ) views for imitation plants 1 in blue and 2 in orange

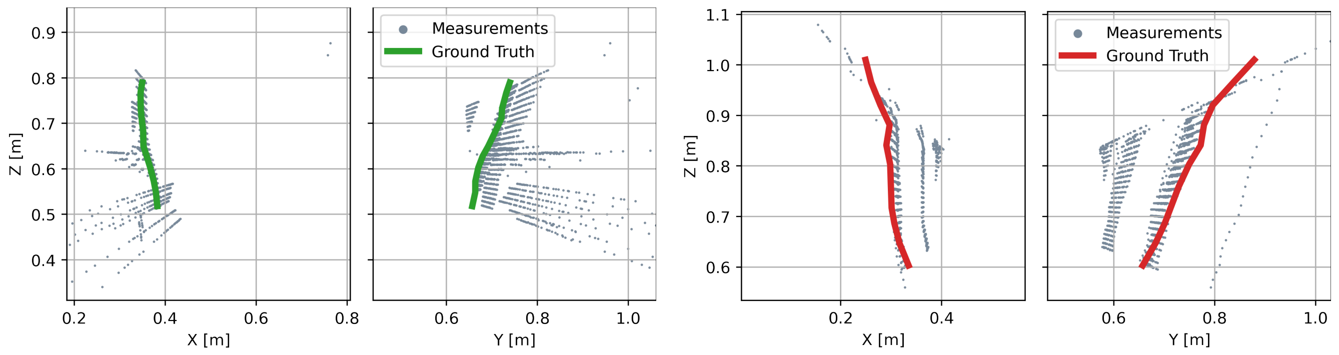
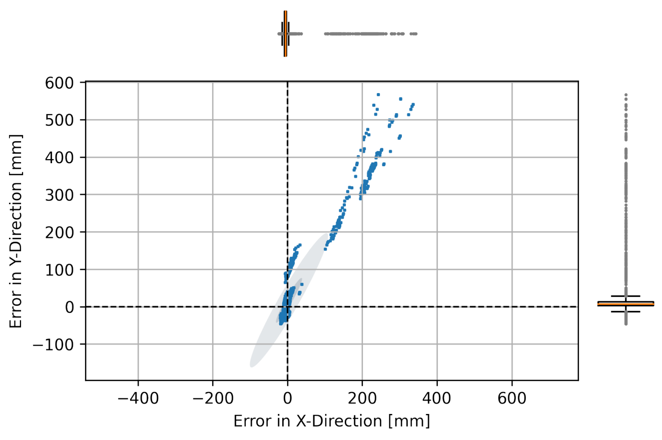


Fig. 25: Front (XZ) and side (YZ) views for imitation plants 3 in green and 4 in red

B.2 Measurement Error to Ground Truth



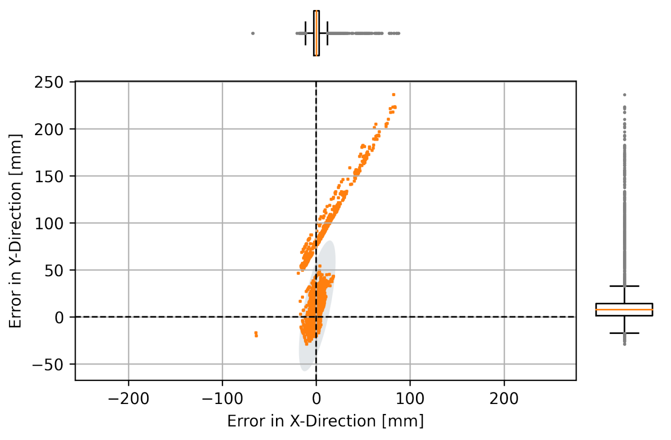
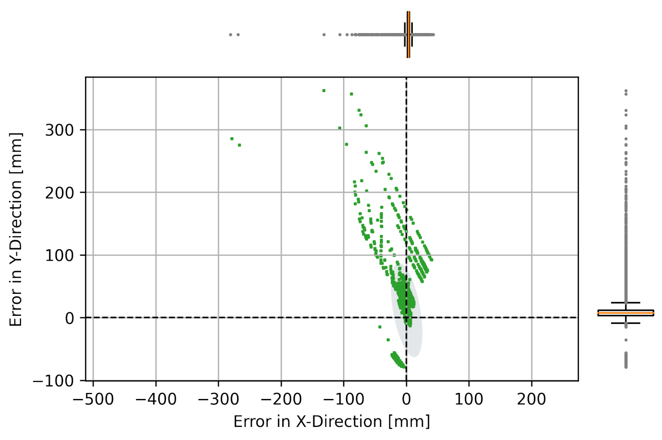


Fig. 26: Plots of error between measurement and ground truth for imitation plants 1 in blue and 2 in orange



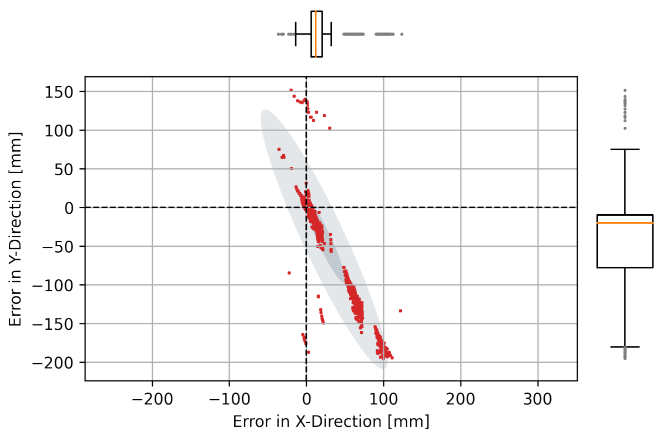


Fig. 27: Plots of error between measurement and ground truth for imitation plants 3 in green and 4 in red

Appendix C
Fixed Plant Structure

C.1 Front and Side Views

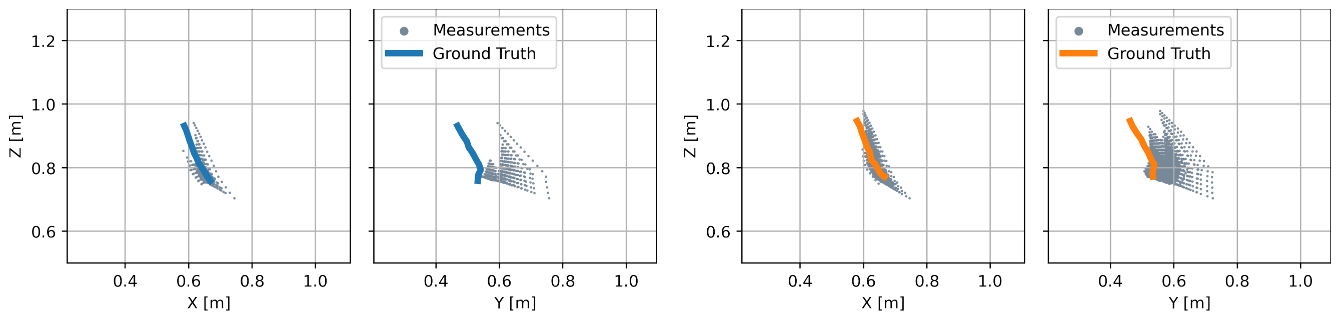


Fig. 28: Front (XZ) and side (YZ) views for fixed plant structure recording Close 1 in orange and 2 in blue

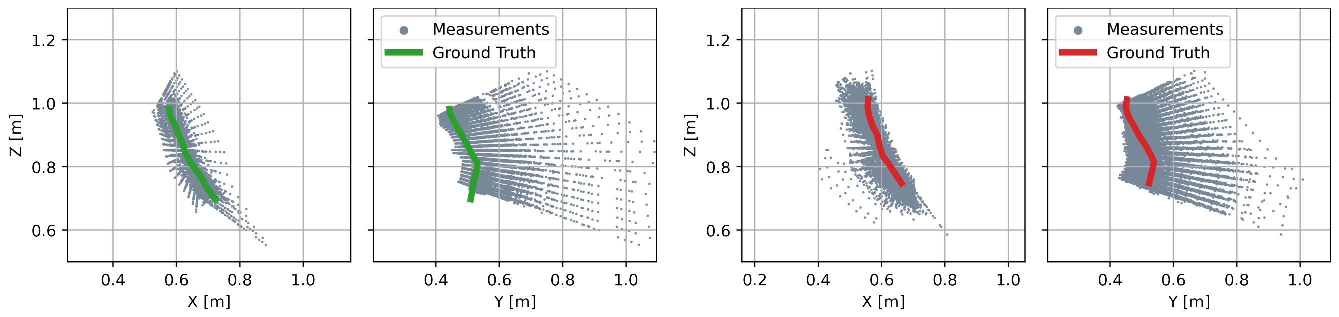


Fig. 29: Front (XZ) and side (YZ) views for fixed plant structure recording Medium 1 in red and 2 in green

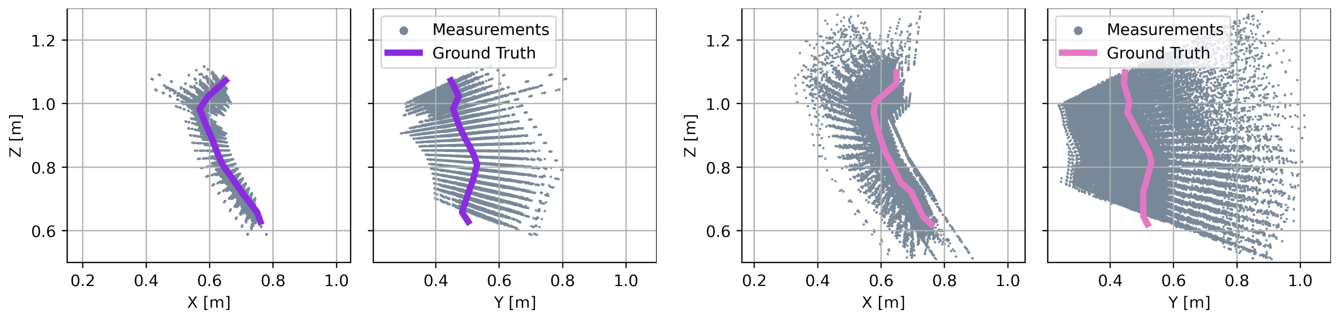


Fig. 30: Front (XZ) and side (YZ) views fixed plant structure recording Far 1 in purple and 2 in pink

C.2 Measurement Error to Ground Truth

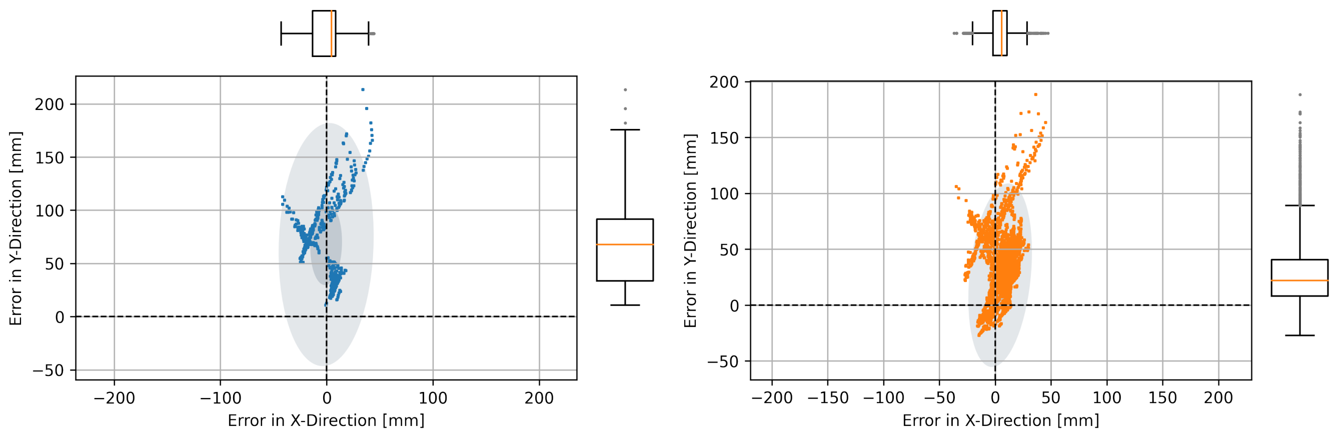


Fig. 31: Plots of error between measurement and ground truth for fixed plant structure recording Close 1 in orange and 2 in blue

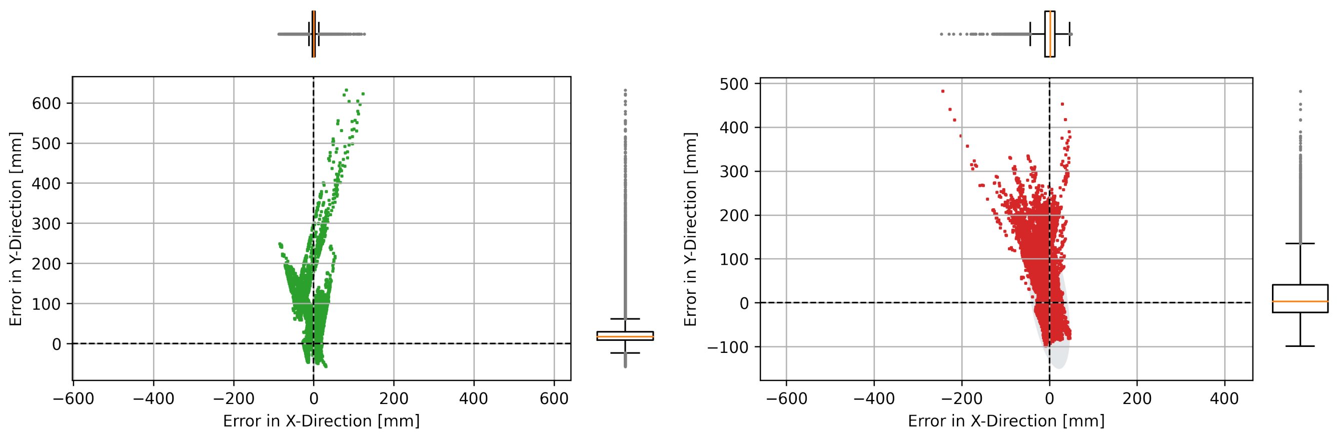


Fig. 32: Plots of error between measurement and ground truth for fixed plant structure recording Medium 1 in red and 2 in green

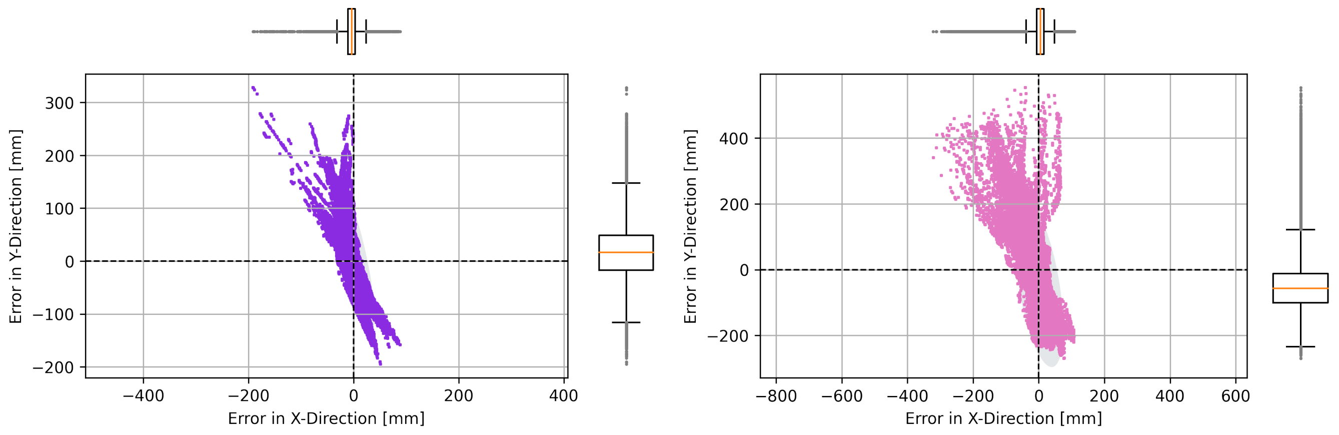


Fig. 33: Plots of error between measurement and ground truth for fixed plant structure recording Far 1 in purple and 2 in pink

TABLE 6: Accuracy for normalized errors in fixed plant structures, all measurements given in mm

	Median		**Mean**		**SD**		**RMSE**
	X	Y	X	Y	X	Y	
Close 1	4.299	67.81	-0.486	67.68	14.88	38.03	79.05
Close 2	5.771	21.95	4.400	26.71	9.487	27.36	39.64
Mid 1	1.022	17.41	0.560	21.98	7.003	25.29	34.24
Mid 2	0.529	3.032	-1.790	12.65	16.24	54.49	58.28
Far 1	-3.818	16.36	-3.445	16.14	14.41	50.67	55.20
Far 2	4.567	-57.44	1.187	-46.69	24.43	83.17	98.47
All	0.716	7.424	-0.452	-2.787	17.17	66.07	68.33

Appendix D
Literature Research

Full literature research can be found on subsequent pages.

A Comprehensive Overview of Depth Sensing in a Greenhouse Environment

Daan Wijffels – 4476247

Abstract—The world's requirement for food increases along with it's population. As a result, there is an increasing demand for automated food production, with greenhouses being specific and hard to tackle example. A depth sensing system is vital for the operation of a robot in this kind of environment. In order to establish comparisons, this paper provides an overview of the available sensors and techniques after which they are scored. Scores are given for resolution, accuracy, speed, field of view, sensor size, compute, and usability in a greenhouse. Due to its excellent resolution, speed, and small size, stereovision was ultimately determined to be the most effective method.

1 Introduction

According to [1, Our World in Data], the current global population is increasing by 1.05 percent, or about 81 million additional individuals year over year. As a result, there is a constantly increasing demand for food, and thus the need to produce food as efficiently as possible. This efficiency can be measured in terms of water and/or energy consumption, as well as the amount labour that is required. Numerous research has been carried out to enhance each of these aspects. We are now able to produce crops with a higher yield, all in thanks to genetic modification of seeds. Water usage has decreased thanks to greenhouse horticulture, which has also made it possible to cultivate vegetables in previously impracticable regions. And finally, mostly all of the world's field-grown vegetables, such as for example potatoes, corn, rice, and low leaf vegetables like spinach or cabbage, can be harvested semi-automatically, and thus require a lot less manual labor. As these crops either have firm fruit or are spread out widely in rows (in the case of low leaf vegetables) they allow for an easier way of automating the harvest and crop care. However many other produce – especially those grown in a greenhouse – still require a great deal of human attention. These vegetables are more delicate, and are mostly farmed vertically. As a result, there has been a push in recent years to automate the greenhouse. For this to be viable, the solution should be able to perform specific tasks like: harvesting of fruit, removal of low hanging leaves as plants grow, pruning of suckers (small offshoots of the plant that hamper it's growth), and especially for tomato plants the repositioning of the plant as it grows. The repositioning of the tomato plants is needed because they grow at around 0.25 meter per week, or about 13 meters in length in a year [2, Dieleman et al.]. For this the stem is wrapped around a small wire, as the plants grows the wire is moved at the top of the greenhouse. This allows the plant to sag as it grows, Figure 1 shows a schematic of the tomato plants in a greenhouse. The repositioning, pruning of suckers and removal of lower leafs has to be done every week, however with current sizes of greenhouses this becomes a consistent task for growers.

State of the art robots as those from [3, Wageningen University] and [4, Chonnam National University] have proven to work, however they are only designed to pick fruit, be that of sweet peppers or tomatoes in the cases mentioned. In order to create a device that can perform all operations necessary in a greenhouse, a system to detect the vegetation is required. This means not only being able to detect and localize fruit but also the ability to generate the structure of the plant, such as its stems and leaves. In order to be useful, the obtained plant structure needs to incorporate a depth component. In most cases this depth data is obtained as a point cloud. Previous research - like that of [5, Neubert et al.] and of [6, Quan et al.] - has even proven that this point cloud can be used to generate a representation of plants and trees. This data is vital for an upcoming robot, as without it moving a machine in 3D space in and around plants would prove to be a lot more challenging. This is where this paper steps in, to provide a comprehensive overview of all the ways that this depth data could be gathered, and to answer the following question:

What is the optimal technique of obtaining depth data for plant structure generation in a greenhouse environment?

As there is a wide range of sensors that can estimate depth we will limit this papers to those that can create a point cloud, thus sensors that meet the following requirements:

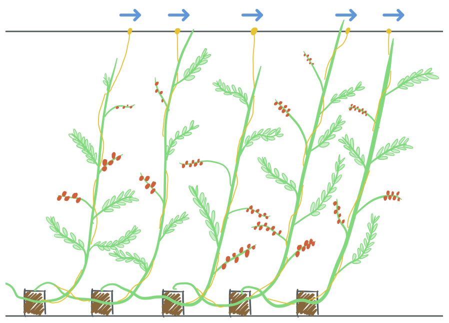


Fig. 1. Schematic visualization of tomato plant growth in a greenhouse, blue arrows showing the direction of repositioning, wrapping wires shown in yellow, and plant shown in green.

- Detect the distance of multiple points
- Have a 2D Field of View (FoV)
- Operate without touching the objects

This will limit this paper to the following sensors and techniques: Radar, LiDAR, Time of Flight Cameras, Stereovision, Structure from Motion, Structured Light and Deep Learning Depth Estimation. The first three sensors on this list are based on the time of flight principle, while the last four are based on computer vision methods using common cameras. The list was compiled with sensors that are in production or are in current research.

This paper is structured as follows; in section 2 and section 3 an overview of the time of flight sensors and camera based techniques are given respectively. In section 4 the sensors will be compared based on the following metrics: Resolution, Accuracy, Speed, Field of View, Size, Cost and Greenhouse Usage. Lastly a conclusion will be given in section 5 to determine the optimal sensor for obtaining plant structures in a greenhouse.

2 Time of Flight Based Sensors

Time of flight is one of the oldest techniques of obtaining a distance measurement. It works by sending out waves or pulses – be that: pressure (sonar), radio (radar) or light (LiDAR) – and timing the return of that wave as it bounces of an object. The sensor therefore consists of two components: an emitter that emits waves or pulses, and an observer or camera that detects the waves or pulses as they return. The distance can be calculated with the following equation:

$$d = \frac{ct}{2} \tag{1}$$

Where d is the distance between our object and sensor in meters, t is the time between send (Tx) and return (Rx) signals in seconds, and c is the wave speed in meters per second. The formula is divided by 2 as we measure the time between sending and return, this means that the wave has traveled twice the distance that we want to measure. See Figure 2 for a visualization of this type of sensor. For sonar, c is the speed of sound, and for radar and LiDAR this is the speed of light. As the speed of sound is highly dependent on the density of the air and thus the air temperature (e.g. $c_{0^\circ C} = 331.4$ m/s and $c_{20^\circ C} = 343.3$ m/s) these readings become highly variable without correct air density measurements. Due to the fact that sound travels farther than electromagnetic waves in water, sonar is more frequently used in water than in air. Next to this the speed of light is at $299,792,458$ m/s orders of magnitude faster than that of sound. This implies that these sensors have significantly lower latency and can sample more points in the same amount of time. We won't continue to look into sonar in this paper as a result.

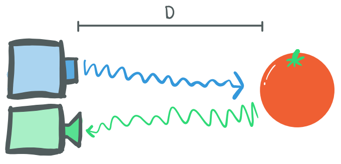


Fig. 2. Time of Flight sensors visualized, blue is the transmitter, green is observer, and D is the distance between object and sensor.

2.1 Radar

Radio Detection and Ranging or as it is better known by it's acronym RaDaR, is a widely used technique of obtaining distance measurements. But has found most of its use cases in military, automotive and naval applications. The first device to detect objects by way of using radio waves was that of Christian Heulsmeyer in 1904 [7, US Patent 810150]. It used a transmitting antenna with broad coverage, and a rotating cylindrical parabolic antenna with a narrow focus. By rotating the receiving antenna the device could detect objects that reflect the waves of the transmitting antenna. The device – which Heulsmeyer called the telemobiloscope – was unfortunately not a true radar as it lacked the ranging aspect. This would come two years later in 1906 by way of a new patent [8, DE Patent 169154] also by Heulsmeyer. This would move the receiving antenna between measurements and use the angle and trigonometry to calculate the distance to the object.

Since this time there have been a lot of improvements made to radar technology, mostly to the way that distance is calculated. Two main techniques namely, Pulsed radar and Frequency-modulated continuous-wave radar (FMCW) have been invented. Both with there own way of solving the distance problem.

Pulsed Radar

The idea of the pulsed radar was first comprised by the RAND Corporation – which stands for the Research and Development Corporation – towards the end of the second world war, and was commissioned by the US Air Force. The theory of the pulsed radar was later unclassified in 1960 [9]. As the name implies, this type of radar uses radio frequency pulses or bursts rather than continuous waves, like the radar of Heulsmeyer. The distance can than be calculated with Equation 1, with t the time between sending and receiving of the radio pulse. However this also means that the maximum and minimum distance that can be measured is related to the frequency at which the pulses are sent. This is because the pulses can not overlap in order for the system to work, we call this the "eclipsing" of the pulses. The maximum and minimum distance can be calculated as follows:

$$D_{\text{min}} = \frac{c\tau}{2}, \quad D_{\text{max}} = c\left(\frac{\text{PRI} - \tau}{2}\right) \tag{2}$$

Where τ is defined as the duration of the pulse and PRI is the Pulse Repetition Interval or the time between the start of two pulses. See Figure 3 for a detailed explanation of these terms. This figure also shows the ambiguity that can occur when we detect an object with a higher interval than the RPI. As we can not determine which Tx pulse generated the orange Rx pulse in the figure, meaning the object could be at 4 different distances for the graph given.
Later research has shown that the speed of the object can also be calculated with a single pulse by using the doppler effect [10]. The doppler effect is a physical phenomenon in where waves interact with a moving object. One known

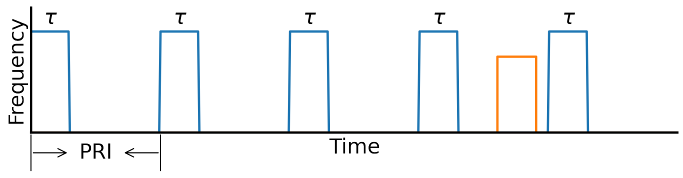


Fig. 3. Visualization of a pulsed radar frequency over time, this figure also demonstrates the pulse receive ambiguity and doppler effect that can occur. Tx pulses given in blue and Rx pulses given in orange.

example of this is the passing by of an ambulance or police car with it's sirens on. As the vehicle moves towards the observer the waves get "bunched" together, making the pitch of the siren higher. When the vehicle moves away the observer the waves get "stretched", thus resulting in a lower pitch. This difference in pitch or frequency can be measured between our Tx and Rx pulses, we call this frequency shift the doppler frequency or f_d. The velocity v of the object can then be calculated with the following equation [10]:

$$f_d = \frac{2v}{\lambda} \quad \text{with} \quad f_d = f_{\text{Rx}} - f_{\text{Tx}} \tag{3}$$

Where λ is defined as the wavelength of Tx pulse. As the plants we aim to detect are stationary, we can use this to detect or correct the speed of the robot. The correction can happen as odometry measurements of robots can differ from real life movement, due to for example slipping of wheels.

Frequency-modulated continuous-wave Radar

Frequency-modulated continuous-wave radar (FMCW radar) uses the same basic principles of object detection as we have seen before. However opposing to pulsed radar; FMCW uses – as the name implies – a continuous wave. Simple continuous wave radar systems that use a single frequency have the downside that they can not detect the objects distance as it lacks a reference mark to compare the Rx and Tx signals to. Such a reference mark can be generated by modulating the frequency of the transmitter, this modulation can be in any form like that of: sawtooth, triangular, rectangular or sine modulation. As sawtooth modulation is the most frequently used [11] we shall use this in rest of this paper. Figure 4 shows a visualization of a sawtooth modulated FMCW radar signal, with Tx and Rx signals.

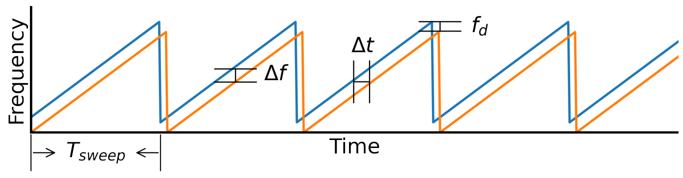


Fig. 4. Visualization of a frequency-modulated continuous-wave radar frequency using sawtooth modulation, this figure also show the beat frequency (Δf), time shift (Δt) and doppler frequency (f_d). Note that Δf and f_d are not the same. Tx given in blue and Rx given in orange.

One of the advantages of this type of radar is that there is no theoretical limit to the distance we can measure as with pulsed radar, and is thus only bound by the polling time of the measurement equipment. The maximum distance for a FMCW radar is determined by the sweep time (T_{sweep}) of the modulation. This is the time between repetition in our signal, see also Figure 4. We calculate the maximum distance for a FMCW radar with the following equation:

$$d_{\text{max}} = c\frac{T_{sweep}}{2} \tag{4}$$

We calculate the velocity of the object the same way as with a pulsed radar, thus using Equation 3. Finally, for a FMCW radar we can calculate the distance to the object with the following equation.

$$d = \frac{c|\Delta t|}{2} = \frac{c|\Delta f|}{2\frac{\mathrm{d}f}{\mathrm{d}t}} \tag{5}$$

Most modern radar systems use the FMCW model as it allows more flexibility in the design of the system. As for example T_{sweep} can be chosen independently of the bandwidth of the system, allowing the designer to modulate at what frequency and thus distance the radar works optimally. Next to this FMCW radars are more accurate than its pulsed counterpart [12], and can perform with cheaper electronics. An FMCW radar does not need an extremely accurate timing device as we measure between our Tx and Rx signals, this can be done with analog to digital converters, which are widely available and at a very low cost.

Steering and Focussing

The radar discussed in the sections above all use the same way of detection the position of an object, by way of using a rotating receiver or transmitter. This technique is still widely used, mostly in naval applications. However this does come with some downsides. First of all it's size and complexity, as adding rotation means adding bearings and a motor to the whole assembly. On top of this if we want to also determine the 3D location –instead of just the angle and distance between the sensor and the object– we also have to rotate the sensor in 2 DoF's instead of 1 making it even more complex. Over the years new ways of steering radio waves have been found, eliminating the need for rotating receivers or transmitters. Which in turn allows for smaller sensors without moving parts. The solution is that of the phased-array radar design [10]. Here the diffraction (also knows as interference) between radio waves of multiple transmitters is used to "steer" the radio waves. For this the transmitters are uniformly a distance d apart, for this distance it is common to take $d = \frac{\lambda}{2}$. As this reduces the complexity of Equation 6 we use to calculate the phase shift.

$$\Delta\phi = \frac{2\pi d \sin(\theta)}{\lambda} \tag{6}$$

The phase shift $\Delta\phi$ is as the name implies a shift in phase between two neighboring transmitters, that is needed to steer the radio wave front an angle θ. This is caused by the fact that with this configuration the phase of the waves of all transmitters line up at this angel, a further visualization of this phenomenon and phased array schematic can be seen in Figure 5. However as we are dealing with waves and their diffraction the waves do not interact in the same way as the visualization shows. As with the visualization the phase along the axis aligned with θ do line up, however as we move away from this axis the waves start to cancel out, forming a beam –or as it also known as a lobe– of radio waves along this axis. We can calculate the diffraction and thus the gain for any configuration of transmitters and steering angle. For this it is common to normalize the calculated lobe gain with the gain of a single transmitter, to create an "outline" of the lobe shape. We call this the array factor or AF of the phased-array. Several examples of different phased-array configurations have been given in Figure 6. These plots have been created with Equation 7 below, with θ being the steering angle and ϕ being the angle at which the AF has to be calculated.

$$AF(\phi) = \frac{\sin\left(\frac{N\pi d}{\lambda}\left[\sin(\phi) - \sin(\theta)\right]\right)}{N\sin\left(\frac{\pi d}{\lambda}\left[\sin(\phi) - \sin(\theta)\right]\right)} \tag{7}$$

From Figure 6 it can be clearly seen that by adding more transmitter to the array we can narrow the main lobe or beam of the array. This is advantageous for our sensor as the smaller the beam, the smaller the detection area. Which in turn increases the theoretical resolution of our sensor. However this does come with one side effect: that of the side lobes, these are smaller less powerful lobes that are created with the interference of the radar array. The gain of these side lobes increases as our steering angle θ increases. Because of this we start detecting more unintended objects as the steering angle increases. Next to this we can also see that the main lobe becomes less narrow as the steering angle increases. Combining this we start seeing the limitations of a sensor using a phased array layout, as we get a significant drop off in detection accuracy and resolution the bigger our field of view becomes. Another aspect to keep in mind is that by adding transmitters our sensor becomes ever more larger, something that is not desired in our optimal sensor.

Ghost Objects

Another problem for radar systems is the fact that radio waves don't just bounce off the target object, but of every object. This is not a problem for the military usecases of detecting planes and or boats, as the objects are sparse in a open space. However when we use radar in a greenhouse this could cause trouble. As a single radio waves can bounce

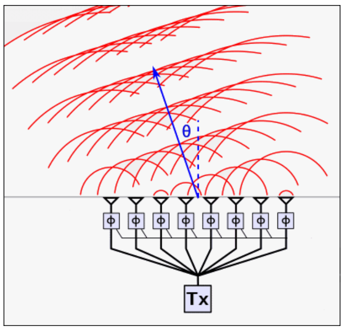


Fig. 5. Schematic visualization of a Phased-Array radar with 8 transmitters, also showing the phase shift and corresponding wave steering angle θ. Visualization does not show the diffraction between waves. [13]

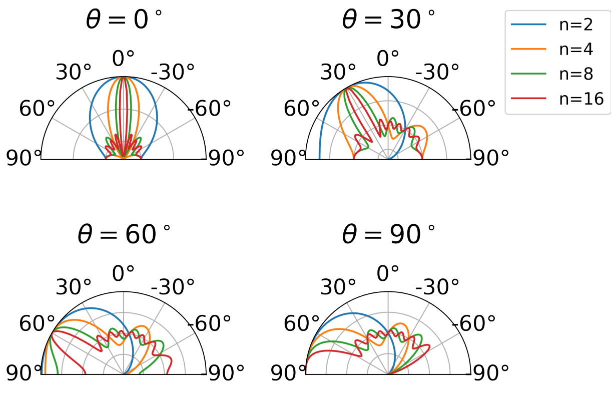


Fig. 6. Plots of Phased-Array Beam Forming Lobe Shapes. Plotted for: $d = \frac{\lambda}{2}$, multiple steering angles θ, and multiple transmitters in the array n ranging between 2 and 16.

of multiple objects back to the sensor, we can measure so called ghost objects, these are objects that we detect but which are not there. This problem is in contrast to the steering problem a lot more difficult to eradicate, and thus extensive research has been done on the issue. Deghosting algorithms by [14], [15], [16], and [17] are among the several that have been developed. For instance in the study by [15, Yant et al.], an algorithm using Bayes' theorem and likelihood function was developed. In the study by [16, Jao et al.], multiple radars are used to determine between real and ghost objects. Finally, the paper by [17, Bekiroglu et al.] shows a algorithm that can determine ghost objects by there trajectory. As ghost objects are created by multiple radio wave bounces, there location fluctuates more than others. The complexity of this trajectory can be calculated and used to distinguish between real and ghost objects. All of these algorithms are based on a hand full of objects that we want to measure, this is something that does not happen in our greenhouse scenario. As the "wall" of leaves and plants in front of the sensor creates an infinite amount of objects that we want to measure. Adding to that, the loose hanging parts of the plants allow for a lot of radio wave reflections and thus results in a lot of ghost objects, something that all above mentioned algorithms are not meant to handle.

State of Art Radar Sensors

Several businesses have started creating so-called imaging radar systems in the last few years, primarily for the automotive sector. The terms mmWave and 4D imaging radar are frequently used to refer to such sensors. Since these sensors are constrained by frequencies designated for the automated industry, the name mmWave is derived from the wavelength of these sensors. This ranges in frequency from 76 to 81 GHz, or a wavelength of 3.95 to 3.70 mm. The second name comes from the fact that these sensors can measure a point in three dimensions while also measuring its speed, creating a four-dimensional measurement. Three manufacturers are found most commonly when buying one of these sensors, that being: Texas Instruments (TI), NXP Semiconductors, and Vayyar Imaging. With the exception of NXP, which also creates its own transmitter and receiver, all of these businesses primarily focus on the development of the Integrated Circuits or IC's of these sensors. Table 1 provides a summary of the technical details for the most cutting-edge sensors produced by these three companies. Data was gathered from each company's website.

TABLE 1
Radar Sensor Comparison

		Texas Instruments AWR1642	NXP TEF82xx	Vayyar XRR
Radar Type	[-]	FMCW	FMCW	FMCW
Transmitters	[-]	8	3	48
Receivers	[-]	24	4	48
Frequency	[GHz]	77	76-81	79
Resolution	[°]	1-2	1	1-2
Horizontal FoV	[°]	70	65	170
Vertical FoV	[°]	40	14	70

By looking at this, we can see that every company uses the FMCW model for its radar, which is consistent with the results of prior research presented in this study. Next to this we also note the correlation between the number of receivers and the FoV of the sensor, with a higher number meaning of receivers corresponding to a higher FoV. One part to note here is the size of the devices, as both the Vayyar and Texas Instruments use an integrated PCB design, and NXP chose to split the IC and transceivers, both parts being extremely small at around the size of a penny. Because of this the NXP chip is designed to work with multiple transceiver units to generate a wider field of view. Table 1 also shows us the biggest downside of using a radar for our greenhouse usecase, that being the resolution. All sensors appear to be bounded by a resolution of 1 degree, this meaning that in the best case – that being the Vayyar sensor – the output depth image is only 170 by 70 pixels. As the plant stems are very narrow this could mean that they could go undetected, or only have a single point of measurement across, making filtering of this data almost impossible.

2.2 LiDAR

LiDAR, also known as Light Detection And Ranging, was created not long after the first laser was developed in 1960. Since methods for determining range by time of flight were developed far earlier. The Hughes Aircraft Company is frequently cited as having developed the first laser-based radar system [18] in 1961. Not by chance, this business was also the first to develop a functional laser, allowing for the development of the LiDAR system. As LiDAR and radar are based on the same time of flight principle, a lot of the developments made for radar can also be used for a LiDAR system. Because of this we have two ways of detecting the distance too an object, that being: Pulsed LiDAR which is more commonly known as a ToF Camera, and Frequency-modulated continuous-wave LiDAR (FMCW LiDAR).

Frequency-modulated continuous-wave LiDAR

A FMCW LiDAR operates in the same was as that of its radar counterpart, by using an frequency swept signals –more commonly called a chirped signal in a LiDAR system– to calculate the distance to an object. For this the sawtooth signal shown in Figure 4 is also used in most applications. Because of this Equation 4 and Equation 5 still hold for a LiDAR system. The big difference is in the way that the beat frequency Δf is calculated, due to the magnitudes higher frequencies that a LiDAR system operates at. Most LiDAR systems work with near visible light, that being either near Infrared (IR) or near Ultraviolet (UV) light. Table 2 shows a comparison of the frequencies used for radar and LiDAR systems. The sampling frequency is determined by the Nyquist-Shannon Sampling Theorem [19], this theory states that in order to measure a specific frequency, the measurement device should sample it at twice the frequency.

TABLE 2
Frequency Comparison for Radar and IR/UV LiDAR

		Radar	IR	UV
Wavelength	[nm]	3.7×10^6	700	400
Frequency	[Ghz]	81	430 000	750 000
Sampling Frequency	[Ghz]	162	860 000	1.5×10^6

From Table 2 it is clear that for an UV LiDAR to function the system should sample at around 10 000 times that of its radar counterpart. Research has shown that it is possible to create a terahertz analog to digital convert necessary for these measurements [20], however they are still experimental and thus extremely cost prohibitive. Because of this, a different method of determining the beat frequency must be developed; interference has provided this method [21]. To be more specific it is due to the heterodyne beat frequency, this is created when two signals with equal amplitude but differing frequency interfere with each other. The combination of the two input signals creates a new signal with an envelope that is half that of the beat frequency. As the beat frequency is given with the following equation:

$$\Delta f = |f_{\text{Tx}} - f_{\text{Rx}}| \tag{8}$$

We can convert these frequencies to a wave with the following equation:

$$f = \cos(2\pi f t) \tag{9}$$

Interference between two signals can be calculated with:

$$\sin\theta_1 \sin\theta_2 = \frac{1}{2}\cos(\theta_1 - \theta_2) - \frac{1}{2}\cos(\theta_1 + \theta_2) \tag{10}$$

The envelope of the function above is given by its first term $\frac{1}{2}\cos(\theta_1 - \theta_2)$. Combining all equations above we get:

$$\frac{1}{2}\cos(2\pi(f_{\text{Tx}} - f_{\text{Rx}})t) \rightarrow \frac{1}{2}(f_{\text{Tx}} - f_{\text{Rx}}) = \frac{1}{2}\Delta f \tag{11}$$

For a visualization of these equations as they relate to each other, see Figure 7.
Since the beat frequency is determined by the difference between the Tx and Rx signals, it is significantly lower than that of the signals its comprised of. As well with the fact that the difference between Tx and Rx –ignoring the doppler effect– is set by the maximum and minimum of our sawtooth signal, means that the frequencies we measure can be designed into the system. Which allows for lower cost electronics and sensors for LiDAR systems.
In order for this heterodyne interference of our signals to happen some special optics have been designed. Comprising of two way mirrors and an optical circulator. The first

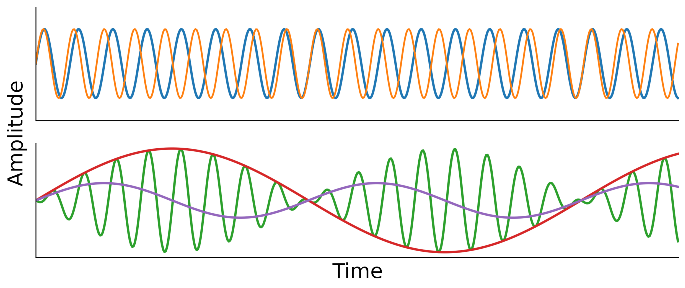


Fig. 7. Visualization of a the heterodyne beat frequency for a LiDAR system. Tx and Rx signals, with Rx being of a larger frequency, shown in blue and orange respectively. Interference between Tx and Rx is shown in green, with corresponding envelope shown in red. Beat frequency Δf shown in purple, note the double period of the envelope as compared to the beat frequency.

two way mirror is a called a splitter, and splits up our signal as the light either reflects or passes through the mirror. The portion of the signal that is kept local and does not travel to the target is called the Local Optical (LO) signal, the second portion traveling to the target is called the Tx signal. The circulator is a passive optical component with three or four ports, designed such that light entering any port exits from the next port. For our LiDAR setup only 3 ports are needed, with our Tx signal entering port 1, it exits the device at port 2. Out of this port the light travels to our target and back to the same port of the circulator. This light now entering port 2 is exited through port 3, effectively splitting our Tx and Rx signals. The Rx and LO signal are sent through the combiner, which is another two way mirror combining the two signals. This final signal is then sent to the detector which converts it to a digital signal which can be converted to a distance. For a overview of this optical setup see Figure 8

LiDAR Steering

As with radar systems, in order to generate a point cloud LiDAR systems have to scan their environment. Most commonly this is done by way of a rotating mirror. As the laser exits the circulator it is reflected of a mirror positioned at 45 degree to the laser, as it rotates the laser is reflected of objects surrounding the LiDAR. This does mean that devices are bigger and more prone to vibrations. Opposing this is are so called solid-state LiDAR systems, here a differentiation can be made between mechanical and non-mechanical solid-state LiDAR. Of the non-mechanical systems the most notable is the phased-array design, this functions the same as with radar. Several of these so-called optical phased-array systems have been created [22], [23], [24], but only those with a maximum FoV of 20 degrees have been successful in research [25]. The efficiency of these systems is another drawback because even the finest ones only achieve 20% efficiency [26]. Most LiDAR manufacturers have shifted to mechanical solid-state devices as a result of these shortcomings. The MEMS mirror-based systems are the most common of these. A microelectromechanical system, or MEMS, is a tiny silicon-based mechanical component created using lithography in a manner similar to that of computer chips. This way of manufacturing also allows for tight knit integration between the mechanics and electronics of such a component. The accelerometer, which is frequently found in smart phones, is a well-known example of a MEMS. Here, a tiny silicon weight is hung from thin silicon members that act as springs. Small combs that are surrounding the weight interact with combs at the sensor's base create capacitors. As the sensor is moved, the distance between these combs varies, affecting the voltage in the capacitors. The MEMS mirror essentially operates in the other way by applying voltage to the combs that move the weight that is linked to the mirror. By angling the mirror the laser beam can be deflected to scan the environment. This can be either done by a single point moving both horizontal and vertical with two MEMS mirrors [27]. It could also be achieved by using specific optics to create a scanning line as shown by [28, Druml et al.], using only one MEMS mirror.

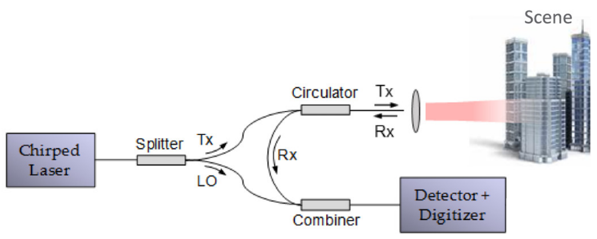


Fig. 8. Visualization of a LiDAR optical system.

2.3 ToF Camera

The Time of Flight Camera (ToF Camera) is an alternative type of solid-state LiDAR. In comparison to prior kinds of LiDAR, this one functions a lot more like a regular camera, as the name implies. Using a CMOS sensor which is commonly found in cameras of all types, for its laser detection [29]. Older research used its analog counter part: the CCD sensor [30]. These kinds of sensors can detect light in a 2D array, allowing the sensor to capture the whole FoV in a single shot. The ToF Camera calculates the distance the same way as previous LiDAR systems by utilizing the phase shift between Tx and Rx signals. This is accomplished by employing a modulated laser signal, which is typically modulated in a square wave [31]. The laser signal is diffused by use of a lens, to light the whole scene at the same time. Detection of this signal is done by a CMOS sensor with two detectors, or "wells", per pixel. By timing the gate of these detectors with that of the laser we can detect the phase difference over time by the accumulated light that hit the detectors. Let us call the two detectors A and B, with A being opened when the laser emits light and vice versa for detector B. As the light gets slightly delayed the returned light is split between the two detectors, this difference can then be used to calculate the phase difference. This also allows the sensor to compensate for differences in reflectance. For highly specular objects – like cars – a technique using different polarizations of light exists [32]. In most systems multiple pulses are used to generate the phase shift, to improve the accuracy of the measurements. A visualization of this light energy over time and split between the detectors can be seen in Figure 9. Because of it's flood illumination and needing multiple pulses to detect distance means that ToF cameras usually have a lower range and higher power usage compared to other LiDAR systems.

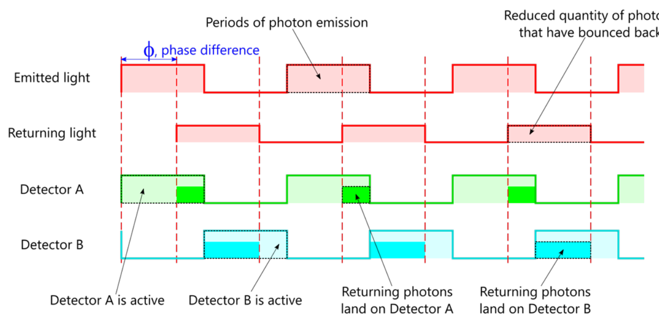


Fig. 9. Plot of light energy over time for a single pixel in a ToF camera. The accumulated difference between the two detectors will be the output of this pixel. [31]

State of Art LiDAR sensors

In the field of LiDAR multiple companies like: Velodyne, Robosense, Sick and Valeo have started producing LiDAR sensors. Most having started of creating rotating mechanical LiDAR's and just recently releasing solid-state versions. For comparison of different technologies the top models from both Velodyne and Robosense have been picked, that being of the rotating and solid-state type respectively. As ToF camera's use a completely different technology the four companies named earlier do not offer any options to compare to. Because of this a sensor from Analog Devices has been chosen to compare too. Comparison of these sensors can be found in Table 3. Looking at these results it can be seen that rotating LiDAR offers a much greater horizontal FoV than it competitors, this is also the reason this type can be found mostly on self driving cars. The MEMS type LiDAR shows great resolution but lacks in accuracy compared to the other two sensors. Finally we can see that the ToF has the best resolution and vertical FoV, both important to detect plant stems. It lacks however in range compared to the other options, however for the distances between plants en sensor in a greenhouse a max range of 3 meters is not a problem.

TABLE 3
LiDAR Sensor Comparison

		Velodyne Ultra Puck	**Robosense** LiDAR M1	**Analog Devices** AD-FXTOF1-EBZ
LiDAR Type	[-]	Rotating	MEMS	ToF
Resolution	[°]	0.33	0.2	0.14
Horizontal FoV	[°]	360	120	87
Vertical FoV	[°]	40	25	69
Range	[m]	0.5-200	0.5-200	0.2-3
Accuracy	[-]	±30mm	±50mm	$\leq 2\%$

3 Camera Based Techniques

With the rise of smaller and better cameras so have the techniques improved to use the data from these sensors. We call the extraction of this data from images Computer Vision. Lots of algorithms have been created in this field and some are designed to detect depth in images. An overview of these techniques is given below.

3.1 Stereovision

As with human eyes a stereovision camera uses 2 camera views to detect depth. We do this by using the pinhole camera model, a mathematical way of modeling light rays through the cameras aperture and onto the image sensor. With this the image is flipped when hitting the sensor, because of this the image is generally modelled at the focal distance in front of the camera aperture. By doing this we can model each pixel on the image as a line starting at the camera aperture, passing through the pixel and ending at the object. A visualization of this can be found in Figure 10, with C1 being the camera aperture and x1 being a pixel on the image. The mathematical formula for converting image pixel or point (x, y) to a line passing through a 3D point (X, Y, Z) is shown in Equation 12, an important variable in this equation is the camera matrix consisting of focal distances f_x and f_y (for a perfect lens these are the same value) and the shift in camera center in both directions c_x and c_y. The camera matrix is unique to each camera, and are approximated by using multiple images of a chessboard with known dimensions and the inverse of Equation 12.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{12}$$

As each image point is modeled as a line we cannot determine the distance to the object with a single camera, as the object could be anywhere along this line. This is where the second camera comes in, as we can use the intersection of two lines passing through the same point on both images to determine depth. To further understand this a visualization of this intersection can be found in Figure 10, here x1 and x2 represent the same object or location on an object on both images.

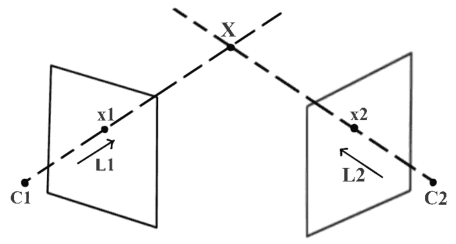


Fig. 10. Visualization of stereovison camera detecting depth at the intersection of lines passing through matched image points x1 and x2. Where X is the object and C1 and C2 are the respective camera apertures of both views. [33]

A common way to calculate matching image points is by using feature matching. Feature matching is a computer vision task all on its own and thus research has provided multiple algorithms to detect and match features on images. [34, Guoshen et al.] shows feature detection using affine transforms and blurring on specific parts of the image, and is commonly known as ASIFT. [35, Alcantarilla et al.] showed a technique called AKAZE, using fast explicit diffusion as apposed to guassian filtering used in ASIFT, in order to speed up calculations. SURF [36], ORB [37], and BRISK [38] are also among the list of feature matching algorithms. Most recently SuperGlue was created using a neural network by [39, Sarlin et al.] outperforming all previous algorithms on accuracy, but taking a hit on speed.

A downside of using feature matching for stereovision is that resulting features always result in a sparse field of matches, in other words we can only calculate the distance for the features that we detect and match between images. Thus leaving a lot of points "empty" in our resulting point cloud. To combat this specially designed algorithms for stereovision were invented, all of them using information of the cameras pose as an extra input to reduce the search space of the algorithms. This is done by using the rectification of the input images. Here the location, rotation and camera matrix of both camera views are used to reproject the images onto a common plane parallel to the line between camera apertures [40]. An example of the rectification of two image views can be found in Figure 11.

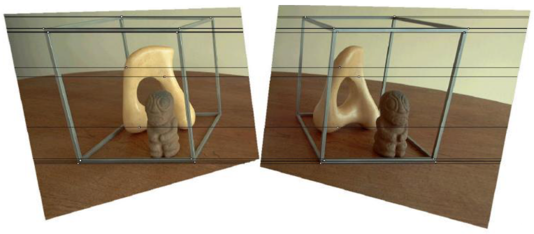


Fig. 11. Rectification of images taken from the same objects but at different locations. Note that after rectification detected features are always horizontally aligned. [40]

The advantage of the rectified image is that resulting image point matches always lay on the same height on both images, we call this epipolar geometry. With this epipolar geometry our search space is thus reduced from the whole image to a single scan line on both images. The most basic form of stereo matching uses a sliding window along horizontal scan lines to find matches. Most often the metric used for finding matches is the sum of squared distances (SSD) together with a small amount of gaussian blur to reduce noise between windows [41]. The formula for the SSD metric can be found below, with W_l and W_r being a window in the left and right image respectively and x and y the coordinates of the pixel in the window.

$$\text{SSD} = \sum_{(x,y)\in W} [W_l(x,y) - W_r(x,y)]^2 \tag{13}$$

By using this metric a best match can be found for each window as it slides over the scan line and create a dense field of matches for our images. This technique is generally called block matching. An important aspect of block matching is the size of the sliding window, a smaller window allows for a higher resolution but introduces noise into the calculations. This can be tuned on a trail and error basis per setting that the stereovision camera is deployed. Improvements to the block matching algorithm have been made by [42, Hirschmuller et al.], with their semi-global matching (SGBM) algorithm. Here mutual information theory is used to improve the matching between blocks, and also address occlusion detection. [43, Amberg et al.] provides another technique called model based stereo, where the image is reprojected onto an approximation of the scene to then refine it with the data that is gathered from the images. In the paper an example of a face is given, using the approximation of a face as an initial guess the tool can generate more detail than with the two images alone. There also exist post processing steps as that of [44, Boykov et al.] which smooth out more noise filled depth data, this can be used to create a better depth image with more resolution and less noise.

One of the main downsides of stereovision is that it relies on texture in images for its block matching and thus depth detection. Objects with little to no texture, like that of a common white wall, have shown to be difficult when it comes to stereovision. In order to combat this most higher end stereo vision cameras employ a infrared dot projector, which can add texture to objects which are hard to detect. Unfortunately these dot projectors are often low power because of their size and power constrains, and thus do not output enough light to be a viable option in a sun lit environment.

3.2 Structure from Motion

Structure from Motion (SFM) is a close cousin to stereovision, using multiple views of a single scene to detect depth. The difference is that SFM uses a single camera at multiple locations compared to the multiple cameras used in stereovision. Because of this SFM is limited to stationary objects, as with a moving camera the capture time between images is not the same. SFM works by using a feature detector and matcher as described in subsection 3.1. Using the matches between images and the pinhole camera model we can use a optimization function (Equation 14) to calculate the location from which the images were taken. Which will allow us to generate a 3D scene from these locations and image points.

$$\underset{\{\mathbf{P}_j\},\{C_i\}}{\text{minimize}} \sum_{i\in j} \left(x_{ij} - \frac{C_{i1}^T \mathbf{P}_j}{C_{i3}^T \mathbf{P}_j} \right)^2 + \left(y_{ij} - \frac{C_{i2}^T \mathbf{P}_j}{C_{i3}^T \mathbf{P}_j} \right)^2 \tag{14}$$

In this equation: C_{ik} denotes the k'th row of the camera matrix, i denotes the camera view, j is a featured matched point in a view, and $\mathbf{P}$ denotes a homogeneous coordinate. This optimization problem is unfortunately non convex and in most cases converges to a local minimum, because of this in most cases other ways of obtaining camera location estimation is used [45]. This can be in the form of, manually capturing the location, using video to observe camera motion [46], or using odometry from a robot. Another constrained that is used is the Cheirality Condition, this says that all constructed points should be in front of the camera. Because the camera matrix equation allows for points to be projected behind the camera, like a mirror at the focal point of the camera. As outliers are a big problem for SfM a bundle adjustment operation is commonly added to the pipeline. Here camera poses are refined by using the reprojection of known 3D points onto 2D images. With these refined camera poses the detection of outliers is greatly simplified. The most commonly used method for bundle adjustment solving is the Levenberg-Marquardt algorithm [47]. The main advantage to SFM is that it can work on any unordered set of images, and no mather the size of the object. Because of this it is often used to capture large objects like buildings or statues. It is also often used when no information is available about the camera or location of the images, think of for example smartphone apps that allow to map a house, or capture objects. The main downsides of SFM is that it takes a large amount of compute to calculate and capture the data, which is why SFM data is normally given in seconds per frame instead of frames per second. One other downside is its reliance on a feature matching algorithm, as previously described this will always result in a sparse point cloud which is undesirable.

3.3 Structured Light

As a light pattern gets shined onto an object this pattern gets distorted by the shape of the object, as the straight rays of light wrap around an object. An example of this distortion and structured light setup can be found in Figure 12. We can use the inverse of this distortion to derive the shape of said object. Distance can be derived from this by size of the pattern. Most often a striped light pattern is used for this application. We can perceive depth on flat, normally featureless objects by using a custom light pattern, which essentially adds our own features to the image. By shifting this pattern from left to right and combining this with a phase unwrapping algorithm [48] to derive the shape and distance of an object. The main advantage of using a phase unwrapping algorithm is due to its excellent accuracy, speed and resolution [49]. Special cases that do not use phase unwrapping have been created, most notably the paper by [50, Cai et al.], here a special light field camera is used to overcome the phase ambiguity. A downside of using structured light is that in order to detect depth the picture has to be sharp, otherwise the lighting pattern becomes blurry and unusable. A paper by [51, Hu et al.] has provided a solution to this problem by using a electronically adjustable lens, allowing to shift the depth of field. By shifting the depth of field multiple images can be used as input to create a sharper image. Another downside is the fact that the structured light technique technique is based on using a pattern of light. Because of this, structured light is often used in light controlled environments. Unfortunately the greenhouse is not one of these environments. Thus requiring an extremely bright light pattern – which would require large amounts of energy – to overcome the light of the sun. There is research done by [52, Gupta et al.] that provides a workflow that can compete with sunlight by focussing the available light on a small part of the image. This also uses around 32 images of the same object from a single location, which would severely decrease the speed of the robot. Another way is to control the sunlight in the greenhouse, thus meaning that the robot could only work at night as sunlight is needed for the growth of the plants. As both of these options severely decrease the speed of the robot the technique of structured light shows to be less desirable than other techniques shown in this paper.

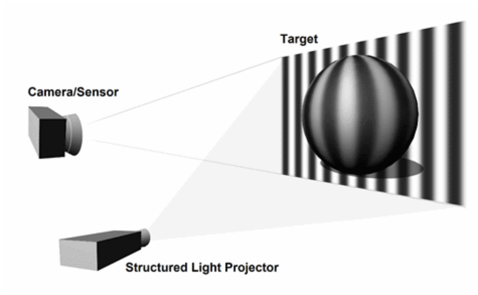


Fig. 12. Setup used in structured light image capturing, and resulting distortion of light pattern. [53]

3.4 Deep Learning

Over the past few years more and more computer tasks have been improved or replaced by the use of deep learning networks. Speech synthesis, image segmentation, object detection, and translation are just a few examples of the usecases of deep learning. In it's most simple form these networks take 2 sets of input x and y, training the network on this data to create y from x. After training the network only takes x as an input and predicts y. Taking object detection as an example, x are images, and y are bounding boxes on set images correlating with the object that we try to detect. This same principle has been used to create depth data from monocular (single camera) images. In this x are the monocular images and y is corresponding depth data. The form of this depth data can differ between networks, however most common are disparity and stereo imaging as an input. Multiple of these so called monocular depth networks have been created, the paper by [54, Zhou et al.] from 2020 gives an excellent overview of these types of networks. Among these networks are those of: [55, Godard et al.] using a CNN with stereo imaging and a custom loss function, [56, Luo et al.] using stereo matching (as described in 3.1) and view prediction, [57, Aleotti et al.] making use of a generative adversarial network (GAN), [58, Pilzer et al.] using a cycled generative network, and finally [59, Kuznietsov et al.] build on direct image alignment loss. These are just a small sample of monocular depth networks that are available, they are mentioned in this paper based on there performance or special architecture. An overview of there performance can be found in Table 4.

TABLE 4
Monocular Depth Neural Network Comparison

	RMSE *Higher is better*	**Accuracy** *Lower is better*
Godard et al. [55]	5.927	0.803
Luo et al. [56]	4.681	0.872
Pilzer et al. [58]	4.656	**0.882**
Aleotti et al. [57]	5.998	0.846
Kuznietsov et al. [59]	**4.621**	0.803

One downside of the use of deep learning networks is their reliance on input data, the more input data there is, the better the fit of the model. Multiple large data sets are available for the training of these networks, but two are most frequently found: KITTI [60] which mostly contains data for road settings and was developed for the self-driving sector, and NYU Depth [61] which primarily contains data for indoor environments. However no such dataset is available for a greenhouse environment, and as the KITTI dataset

TABLE 5
Sensor Comparison

	Resolution (2)	**Accuracy** (2)	**Speed** (2)	**FoV** (3)	**Size** (1)	**Compute** (2)	**Greenhouse** (3)	**Total**
Radar	1	3	2	**5**	3	**4**	4	44
LiDAR	3	**4**	2	2	2	**4**	**5**	49
ToF Camera	3	**4**	3	3	3	**4**	**5**	55
Stereovision	**5**	3	**5**	3	4	3	3	**57**
Structure from Motion	**5**	3	1	4	**5**	2	2	48
Structured Light	**5**	3	**5**	3	4	3	1	51
Deep Learning	**5**	2	3	3	**5**	1	1	45

alone holds about 13 thousand images, the task of creating one would be very time consuming. Next to this is the fact that these networks can only approximate depth from input data implies that a different type of system has to be used to capture this data. Combining these two drawbacks raises the question of whether employing a deep learning network is more advantageous than utilizing a larger sensor instead of the single camera.

4 Comparison

Now that we've covered the sensors and how they function, it's time to compare them. For this we will consider the FMCW Radar, MEMS LiDAR, ToF Camera, Stereovision, Structure from Motion, Structured Light and Monocular Depth Deep Learning techniques. The decision was made to not consider Pulsed Radar, as this technique is almost unused in currently available radar systems. It was also decided not to compare rotating LiDAR systems as their field of view and resolution is less compatible with the usecases for our greenhouse robot, when compared to the MEMS LiDAR.

Criteria

A set of criteria has been created in order to evaluate different depth sensing methods. Weights between 1 and 3 have been assigned to each criteria, see Table 5 for weights in brackets behind the criteria. Each criteria will be scored from 1 to 5, the higher the better. The final score will be calculated by the sum of of score with weight multiplication for each technique. Thus the higher the final score the better the technique is for our usecase. Below a list of all criteria with an explanation can be found.

- **Resolution:** Since plant stems can be rather thin, a high sensor resolution is preferred. Resolution is calculated as the angle between measurement points, as this can differ for different field of views. The resolution is higher and the score is higher the narrower this angle is.
- **Accuracy:** High precision is required since we want to measure the separation between the sensor and the plants. Estimates have been made for sensors where accuracy metrics were not available.
- **Speed:** The robot should move quickly in order to have a high work rate. In turn, this means that sensors should have fast polling rates, or high frames per second. The higher the fps the higher the score.
- **Field of View:** Getting as much of the plant in view as possible is desired for the sensors. Most importantly when the robot moves close to the plants a bigger FoV will allow the robot to see more of its surroundings. Scores will be given by multiplying the horizontal and vertical FoV angles. The bigger the FoV the higher the score.
- **Size:** As the intended location of the sensor is on the robot arm size matters as the robot moves between plants. A smaller size of the sensor is thus desired.
- **Compute:** The robot is made to function on battery power, high amounts of compute use a lot of this battery power. In order to keep costs and energy consumption low, lower amounts of compute are desired per sensor.
- **Greenhouse:** As the sensors will be used in a greenhouse, they have to function optimally in this setting. As this metric is highly subjective the reasoning for the score can be found in the section below.

Scores

All scores can be found in Table 5. Scores for most of the criteria are objective and are rated high to low for all sensors. Unfortunately this is not the case for the accuracy and greenhouse criteria. Accuracy is not provided by every manufacturer or researcher. LiDAR and ToF Camera have been given a score of 4, as accuracy specifications are provided by manufacturer. A score of 5 was not given here as accuracy is still high at 2% or 50mm for these sensors, which is deemed high for a greenhouse setting. Radar was scored at 3 as ghost objects degrade its accuracy when compared to its light based counterparts. As stereovision, structure from motion, and structured light are all camera based they are scored the same with a score of 3. Deep Learning was scored lowest as it is an approximation of other sensors, and thus the accuracy will always be lower.

The greenhouse usage metric is highly subjective to the researcher. Structured light and deep learning are scored the lowest, at a score of 1. This is because of its reliance on light patterns, and large currently unavailable datasets respectively. This makes them both very unsuitable to current implementation in greenhouses. We do note that structured light could provide an option in vertical farms, as this is already a completely light controlled environment. Structure form motion and stereovision, do not have the same downfalls and are thus scored higher. Maximum scores have not been given here as both techniques rely on computer vision to solve the depth. Both feature matching and semi-global matching rely on color and texture in images to find matches between multiple images. These methods do not work as well in a greenhouse as they do in other environments due to the lack of color variation, as almost everything in view is green, and the high plant density, which produces a noise-like texture. As stereovision does not rely on camera movement it is scored higher then structure from motion. LiDAR, ToF camera, and radar do not rely on computer vision techniques, and are thus scored higher. Radar has been deducted 1 point because of ghosting objects, this problem is exaggerated in a greenhouse as the sound waves bounce of all the leaves and stems in view.

5 Conclusion

Table 5 shows that, of all the sensors, stereovision has the greatest overall score, and is thus the most optimal sensor for use in a greenhouse. Scoring highly on resolution, speed, and size. The biggest downside of this technique is its reliance on computer vision. As with the images coming from the greenhouse the semi-global matching performs less then optimal. Either creating sparse point clouds or needing higher amounts om compute which consume more energy, and or lower the frames per second of the sensor. The use of a neural network to improve this technique could prove as an interesting point of research, as great improvements to for example feature detection have been made in this field. The ToF camera has shown a good alternative to the stereovision's downsides. If the lack in resolution, speed and size can be improved or overcome in the design of the robot, the ToF camera could overtake stereovision as the most optimal sensor.

References

[1] H. R. Max Roser and E. Ortiz-Ospina, "World population growth," *Our World in Data*, 2013. [Online]. Available: https://ourworldindata.org/world-population-growth

[2] J. A. Dieleman, A. de Gelder, B. A. Eveleens, A. Elings, J. Janse, P. Lagas, T. Qian, J. W. Steenhuizen, and E. Meinen, "Tomaten telen in een geconditioneerde kas: groei, productie en onderliggende processen," Wageningen UR Glastuinbouw, Bleiswijk, Tech. Rep. 633, 2009. [Online]. Available: https://library.wur.nl/WebQuery/wurpubs/383473

[3] B. Arad, J. Balendonck, R. Barth, O. Ben-Shahar, Y. Edan, T. Hellström, J. Hemming, P. Kurtser, O. Ringdahl, T. Tielen, and B. van Tuijl, "Development of a sweet pepper harvesting robot," *Journal of Field Robotics*, vol. 37, no. 6, pp. 1027–1039, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21937

[4] J. Jun, J. Kim, J. Seol, J. Kim, and H. I. Son, "Towards an Efficient Tomato Harvesting Robot: 3D Perception, Manipulation, and End-Effector," *IEEE Access*, vol. 9, pp. 17 631–17 640, 2021.

[5] B. Neubert, T. Franken, and O. Deussen, "Approximate image-based tree-modeling using particle flows," in *ACM SIGGRAPH 2007 Papers*, ser. SIGGRAPH '07. New York, NY, USA: Association for Computing Machinery, Jul. 2007, pp. 88–es. [Online]. Available: https://doi.org/10.1145/1275808.1276487

[6] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang, "Image-based plant modeling," in *ACM SIGGRAPH 2006 Papers*, ser. SIGGRAPH '06. New York, NY, USA: Association for Computing Machinery, Jul. 2006, pp. 599–604. [Online]. Available: https://doi.org/10.1145/1179352.1141929

[7] C. Huelsmeyer, "Wireless transmitting and receiving mechanism for electric waves," US Patent US810 150A, Jan., 1906. [Online]. Available: https://patents.google.com/patent/US810150/en?oq=US810150

[8] C. Heulsmeyer, "Verfahren zur Bestimmung der Entfernung von metallischen Gegenständen," DE Patent DE169 154C, Apr., 1906. [Online]. Available: https://patents.google.com/patent/DE169154C/de?oq=DE+169154

[9] J. Marcum, "A statistical theory of target detection by pulsed radar," *IRE Transactions on Information Theory*, vol. 6, no. 2, pp. 59–267, Apr. 1960.

[10] T. Jeffrey, *Phased-array radar design: application of radar fundamentals*, ser. Radar, Sonar & Navigation. Raleigh, NC: SciTech Pub., 2009. [Online]. Available: http://www.books24x7.com/marc.asp?bookid=75093

[11] F. C. Cruz, "Radar system based on Frequency-Modulated Continuous Wave transmission," p. 59.

[12] Hawk Measurement Systems, "What is the Difference Between Frequency- Modulated Continuous-Wave (FMCW) and Pulsed Wave or Pulsed Width Radar?" [Online]. Available: https://www.automation.com/getattachment/d201a032-1c4b-4885-b41b-b2c0400b3cd2/FMCW-vs-Pulse-Radar-White-Paper.pdf?lang=en-US&ext=.pdf

[13] Wikimedia Commons. (2016) Phased array animation with arrow. [Online]. Available: https://commons.wikimedia.org/wiki/File%3aLambdaPlaques.jpg

[14] S. S. Blackman, *Multiple-Target Tracking with Radar Applications*, Jan. 1986. [Online]. Available: https://ui.adsabs.harvard.edu/abs/1986ah...bookQ....B

[15] R. Yang and G. W. Ng, "Deghosting in multipassive acoustic sensors," in *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2004*, vol. 5434. SPIE, Apr. 2004, pp. 187–194. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/5434/0000/Deghosting-in-multipassive-acoustic-sensors/10.1117/12.538724.full

[16] J. Jao, "Coherent multilateral radar processing for precise target geolocation," in *2006 IEEE Conference on Radar*, Apr. 2006, pp. 6 pp.–.

[17] K. Bekiroglu, M. Ayazoglu, C. Lagoa, and M. Sznaier, "An Efficient Approach to the Radar Ghost Elimination Problem," Jul. 2016.

[18] "New Radar System," *Odessa American*, Feb. 1961.

[19] C. Shannon, "Communication in the Presence of Noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, Jan. 1949. [Online]. Available: http://ieeexplore.ieee.org/document/1697831/

[20] Z. Xu and P. Mazumder, "Terahertz Beam Steering With Doped GaAs Phase Modulator and a Design of Spatial-Resolved High-Speed Terahertz Analog-to-Digital Converter," *IEEE Transactions on Electron Devices*, vol. 61, no. 6, pp. 2195–2202, Jun. 2014.

[21] P. F. McManamon, *LiDAR Technologies and Systems*. Bellingham, Washington, USA: SPIE Press, 2019.

[22] S. A. Miller, Y.-C. Chang, C. T. Phare, M. C. Shin, M. Zadka, S. P. Roberts, B. Stern, X. Ji, A. Mohanty, O. A. Jimenez Gordillo, U. D. Dave, and M. Lipson, "Large-scale optical phased array using a low-power multi-pass silicon photonic platform," *Optica*, vol. 7, no. 1, p. 3, Jan. 2020. [Online]. Available: https://opg.optica.org/abstract.cfm?URI=optica-7-1-3

[23] M. J. R. Heck, "Highly integrated optical phased arrays: Photonic integrated circuits for optical beam shaping and beam steering," *Nanophotonics*, vol. 6, no. 1, pp. 93–107, Jan. 2017. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/nanoph-2015-0152/html

[24] K. V. Acoleyen, H. Rogier, and R. Baets, "Two-dimensional optical phased array antenna on silicon-on-insulator," *Optics Express*, vol. 18, no. 13, pp. 13 655–13 660, Jun. 2010. [Online]. Available: https://opg.optica.org/oe/abstract.cfm?uri=oe-18-13-13655

[25] C. V. Poulton, A. Yaacobi, D. B. Cole, M. J. Byrd, M. Raval, D. Vermeulen, and M. R. Watts, "Coherent solid-state LIDAR with silicon photonic optical phased arrays," *Optics Letters*, vol. 42, no. 20, pp. 4091–4094, Oct. 2017. [Online]. Available: https://opg.optica.org/ol/abstract.cfm?uri=ol-42-20-4091

[26] P. McManamon, "An overview of optical phased array technology and status," in *Liquid Crystals: Optics and Applications*, vol. 5947. SPIE, Sep. 2005, pp. 152–161. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/5947/59470I/An-overview-of-optical-phased-array-technology-and-status/10.1117/12.631412.full

[27] T. Raj, F. H. Hashim, A. B. Huddin, M. F. Ibrahim, and A. Hussain, "A Survey on LiDAR Scanning Mechanisms,"

Electronics, vol. 9, no. 5, p. 741, May 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/5/741

[28] N. Druml, I. Maksymova, T. Thurner, D. van Lierop, M. Hennecke, and A. Foroutan, *1D MEMS Micro-Scanning LiDAR*, Sep. 2018.

[29] I. Vornicu, R. Carmona-Galán, and A. Rodríguez-Vázquez, "Photon counting and direct ToF camera prototype based on CMOS SPADs," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[30] Y. He, B. Liang, Y. Zou, J. He, and J. Yang, "Depth Errors Analysis and Correction for Time-of-Flight (ToF) Cameras," *Sensors*, vol. 17, p. 92, Jan. 2017.

[31] Oliver, "Understanding Indirect ToF Depth Sensing," Apr. 2021. [Online]. Available: https://devblogs.microsoft.com/azure-depth-platform/understanding-indirect-tof-depth-sensing/

[32] T. Yoshida, V. Golyanik, O. Wasenmuller, and D. Stricker, "Improving Time-of-Flight Sensor for Specular Surfaces with Shape from Polarization," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. Athens: IEEE, Oct. 2018, pp. 1558–1562. [Online]. Available: https://ieeexplore.ieee.org/document/8451357/

[33] "Introduction to Epipolar Geometry and Stereo Vision — LearnOpenCV #," Dec. 2020. [Online]. Available: https://learnopencv.com/introduction-to-epipolar-geometry-and-stereo-vision/

[34] G. Yu and J.-M. Morel, "ASIFT: An Algorithm for Fully Affine Invariant Comparison," *Image Processing On Line*, vol. 1, pp. 11–38, Feb. 2011. [Online]. Available: https://www.ipol.im/pub/art/2011/my-asift/?utm_source=doi

[35] P. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces," in *Procedings of the British Machine Vision Conference 2013*. Bristol: British Machine Vision Association, 2013, pp. 13.1–13.11. [Online]. Available: http://www.bmva.org/bmvc/2013/Papers/paper0013/index.html

[36] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *Computer Vision – ECCV 2006*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer, 2006, pp. 404–417.

[37] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*. Barcelona, Spain: IEEE, Nov. 2011, pp. 2564–2571. [Online]. Available: http://ieeexplore.ieee.org/document/6126544/

[38] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," in *2011 International Conference on Computer Vision*. Barcelona, Spain: IEEE, Nov. 2011, pp. 2548–2555. [Online]. Available: http://ieeexplore.ieee.org/document/6126542/

[39] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning Feature Matching With Graph Neural Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4938–4947. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Sarlin_SuperGlue_Learning_Feature_Matching_With_Graph_Neural_Networks_CVPR_2020_paper.html

[40] C. Loop and Zhengyou Zhang, "Computing rectifying homographies for stereo vision," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. Fort Collins, CO, USA: IEEE Comput. Soc, 1999, pp. 125–131. [Online]. Available: http://ieeexplore.ieee.org/document/786928/

[41] Y. Lin, Y. Gao, and Y. Wang, "An Improved Sum of Squared Difference Algorithm for Automated Distance Measurement," *Frontiers in Physics*, vol. 9, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphy.2021.737336

[42] H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, Feb. 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4359315/

[43] B. Amberg, S. Romdhani, A. Blake, T. Vetter, and A. Fitzgibbon, "Reconstructing High Quality Face-Surfaces using Model Based Stereo."

[44] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," p. 8.

[45] O. Ozyesil, V. Voroninski, R. Basri, and A. Singer, "A Survey of Structure from Motion," May 2017. [Online]. Available: http://arxiv.org/abs/1701.08493

[46] M. Naito, K. Matsumoto, K. Hoashi, and F. Sugaya, "Camera Motion Detection using Video Mosaicing," in *2006 IEEE International Conference on Multimedia and Expo, ICME 2006 - Proceedings*, vol. 2006, Jul. 2006, pp. 1741–1744.

[47] J. L. Schonberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 4104–4113. [Online]. Available: http://ieeexplore.ieee.org/document/7780814/

[48] K. Itoh, "Analysis of the phase unwrapping algorithm," *Applied Optics*, vol. 21, no. 14, pp. 2470–2470, Jul. 1982. [Online]. Available: https://opg.optica.org/ao/abstract.cfm?uri=ao-21-14-2470

[49] D. Malacara, Ed., *Optical Shop Testing*, 3rd ed., ser. Wiley Series in Pure and Applied Optics. Hoboken, N.J: Wiley-Interscience, 2007.

[50] Z. Cai, X. Liu, G. Pedrini, W. Osten, and X. Peng, "Structured-light-field 3D imaging without phase unwrapping," *Optics and Lasers in Engineering*, vol. 129, p. 106047, Jun. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014381661931351X

[51] X. Hu, G. Wang, Y. Zhang, H. Yang, and S. Zhang, "Large depth-of-field 3D shape measurement using an electrically tunable lens," *Optics Express*, vol. 27, p. 29697, Oct. 2019.

[52] M. Gupta, Q. Yin, and S. K. Nayar, "Structured Light in Sunlight," in *2013 IEEE International Conference on Computer Vision*. Sydney, Australia: IEEE, Dec. 2013, pp. 545–552. [Online]. Available: http://ieeexplore.ieee.org/document/6751177/

[53] "What is Structured Light Imaging? — RoboticsTomorrow." [Online]. Available: https://roboticstomorrow.com/article/2018/04/what-is-structured-light-imaging/11821

[54] C. Zhao, Q. Sun, C. Zhang, Y. Tang, and F. Qian, "Monocular Depth Estimation Based On Deep Learning: An Overview," *Science China Technological Sciences*, vol. 63, no. 9, pp. 1612–1627, Sep. 2020. [Online]. Available: http://arxiv.org/abs/2003.06620

[55] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," Apr. 2017. [Online]. Available: http://arxiv.org/abs/1609.03677

[56] Y. Luo, J. Ren, M. Lin, J. Pang, W. Sun, H. Li, and L. Lin, "Single View Stereo Matching," Mar. 2018. [Online]. Available: http://arxiv.org/abs/1803.02612

[57] F. Aleotti, F. Tosi, M. Poggi, and S. Mattoccia, "Generative Adversarial Networks for Unsupervised Monocular Depth Prediction," in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Cham: Springer International Publishing, 2019, vol. 11129, pp. 337–354. [Online]. Available: http://link.springer.com/10.1007/978-3-030-11009-3_20

[58] A. Pilzer, S. Lathuilière, N. Sebe, and E. Ricci, "Refine and Distill: Exploiting Cycle-Inconsistency and Knowledge Distillation for Unsupervised Monocular Depth Estimation," Apr. 2019. [Online]. Available: http://arxiv.org/abs/1903.04202

[59] Y. Kuznietsov, J. Stückler, and B. Leibe, "Semi-Supervised Deep Learning for Monocular Depth Map Prediction," May 2017. [Online]. Available: http://arxiv.org/abs/1702.02706

[60] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913491297

[61] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *Computer Vision – ECCV 2012*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7576, pp. 746–760. [Online]. Available: http://link.springer.com/10.1007/978-3-642-33715-4_54