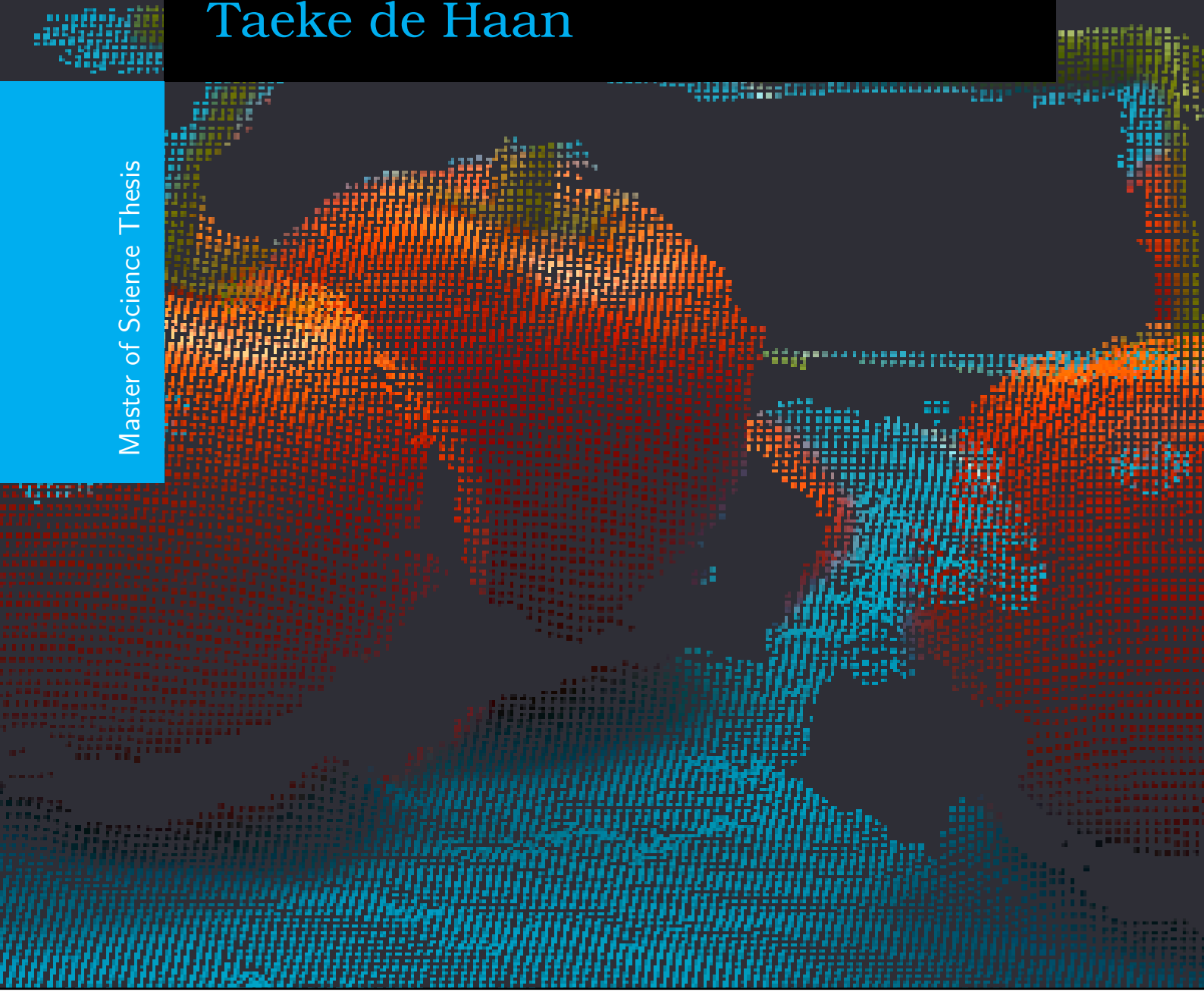
A Geometry-Based Grasping Method for Vine Tomato

MSc Thesis

Taeke de Haan

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control and Cognitive Robotics

A Geometry-Based Grasping Method for Vine Tomato

MSc Thesis

Master of Science Thesis

For the degree of Master of Science in Systems and Control and Mechanical Engineering at Delft University of Technology

Taeke de Haan

October 12, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft Center for
Systems and Control

Cognitive
Robotics

Summary

Vine tomato is the most popular cultivated crop in the Dutch greenhouse horticulture, taking up 21% of the national production surface area. Automating the harvesting and post-harvesting of this crop by employing robotics may diminish the impact of labor shortages, reduce the financial risk for farmers, and increase agricultural efficiency. One of the challenges is how to handle these fragile organic objects without damaging the crop. A geometry-based grasping method is proposed which does not require delicate contact sensors or complex mechanical models.

To limit the object's free path without requiring tight contact, caging is used. The truss stem exhibits various fork features, which are described by a curve with split ends. A loop is formed by the end effector around the stem at these locations to prevent the truss from moving arbitrarily far away. Soft parts placed on the inside of the end effector deform and apply a reaction force to the truss stem. A target grasp location is determined based on simple geometric models of both the truss and end effector. The used criteria take into account the free space on the stem and the center of mass location.

A computer vision pipeline is developed to identify the required geometric features of a given truss. The image is segmented by applying k-means clustering on the hue and a* color components. Tomatoes are detected using the Circle Hough Transform. The stem segment is analyzed using a graph search method, the longest path with limited curvature is said to be the truss stalk. On the constructed test set, 100% of the tomatoes are detected. Prediction errors on the tomato locations and dimensions resulted in an error of 5.0 ± 3.26mm on the truss center of mass. The algorithm predicted 86% of the nodes where the stem splits, but also 34% false positives were reported.

Real-world grasping experiments were conducted using an RGB-D camera and a robotic manipulator. Trusses of various shapes and sizes were constructed by combining plastic and real stems with artificial tomatoes. The newly developed method was shown to be capable of grasping vine tomato. A success rate of 92% was obtained on a truss with an artificial stem and 80% on a truss with a real stem. For future research, it is recommended to incorporate three-dimensional data into the truss model such that the space around the truss stalk can be analyzed. For practical implementation, it is recommended to conduct experiments on larger batches of trusses and extend the method to deal with touching and overlapping instances.

Table of Contents

Acknowledgements

I would like to thank Prof.dr. Robert Babuska from the Delft University of Technology for his assistance for finding an interesting topic, for recommending relevant literature, and for providing feedback on this thesis. Furthermore, I would like to thank Padmaja Kulkarni from the Delft University of Technology for providing relevant literature, giving continuous feedback during the writing process, and clarifying difficult topics. I would like to thank J. Luijkx for assisting me in the lab and during our frequent coffee breaks. Finally, I would like to thank V. Beekman, J. Steijn, and D. Boonstra for performing a thorough spelling and grammar check on this thesis.

Delft, University of Technology — Taeke de Haan
October 12, 2020

Glossary

List of Acronyms

3mE	Mechanical, Maritime and Materials Engineering
PCA	principle component analysis
RANSAC	Random sample consensus
CHT	Circle Hough Transform
RGB	red, green and blue
HSV	hue, saturation and value
HSI	hue, saturation and intensity
HSL	hue, saturation and lightness
YIQ	luminance, in-phase and quadrature-phase
IR	infra-red
ROS	Robot Operating System

List of Symbols

Evaluation

FDR	The false discovery rate
FN	The number of false negative predictions
FP	The number of false positive predictions
TP	The number of true positive predictions
TPR	The true positive rate

Geometric Model Parameters

$\boldsymbol{c}_i$	Tomato center
$\boldsymbol{c}_{\text{com}}$	Center of mass
h_{tip}	Fingertip height
l	End effector length
r_i	Tomato radius
t_{tip}	Fingertip thickness
w	End effector width

Control

c	Safety factor
d_{tip}	Controllable distance between end effector fingers
l_{thresh}	Distance threshold from the junctions encapsulating the branch

Computer Vision Variables

α	The angle between the truss and the horizontal axis of the image
$\bar{\boldsymbol{a}}^*$	Normalized a* vector containing downsampled data.
$\boldsymbol{a}^*$	A* vector containing downsampled data.
$\boldsymbol{h}$	Hue vector containing downsampled data.
a^*	A* color component from the L*a*b* color space
B	Blue color component
C	Chrome color component.
G	Green color component
H	Hue color component
M	Maximum color component.
m	Minimum color component.
R	Red color component
X	X color component from the XYZ color space
Y	Y color component from the XYZ color space
Z	Z color component from the XYZ color space

Image frames

$\mathcal{C}$	Cropped image frame
$\mathcal{I}$	Original image frame
$\mathcal{R}$	Rotated image frame

Computer Vision Parameters

α_{thresh}	Peduncle curvature threshold
ϵ	K-means center movement stopping criterion.
σ_{blur}	Gaussian kernel standard deviation
c_1	Gradient threshold for accepting a pixel as an edge
c_2	The accumulator threshold for the circle centers at the detection stage
c_{fill}	Minimum required overlap between candidate circle and tomato segment
c_{dp}	The inverse ratio between accumulator resolution and image resolution
d_{blur}	Gaussian kernel size
d_{kernel}	Circular kernel diameter in pixels
d_{max}	Maximum amount of edges in search path.
d_{min}	Minimum tomato center distance in millimeters
i_{max}	K-means iterations stopping criterion.
l_{min}	Minimum branch length in millimeter
r	Radius of hue circle
r_{max}	Maximum tomato radius in millimeters
r_{min}	Minimum tomato radius in millimeters

Chapter 1

Introduction

Tomato is the second most produced vegetable crop after potato according to the Food and Agriculture Organization of the United Nations Statistics Database (FAOSTAT) [1]. They are popular for both fresh and processed consumption [1]. Tomatoes are grown in the open field, or if required, in the protected environment of a greenhouse. Vine, truss, and clustered tomato are marketing terms used for tomatoes when kept attached to the fruiting stem after the harvest. The stem provides a characteristic tomato aroma and has a positive effect on the visual appearance of the produce [1]. Vine tomato is the most popular cultivated crop in the Dutch greenhouse horticulture, taking up 21% of the national production surface area [2]. A schematic drawing of a typical tomato truss is shown in Figure 1-1. The tomatoes are attached to the stem at the calyx, which is the flower-like structure. These are connected via the fruit stalk, or pedicels, to the larger truss stalk, or peduncle.

While the harvesting and post-harvest handling of processed tomatoes is generally performed mechanically, preparation of fresh tomatoes relies on skilled workers [1]. Vine tomato is

[1] Public database http://www.fao.org/faostat/en/#compare

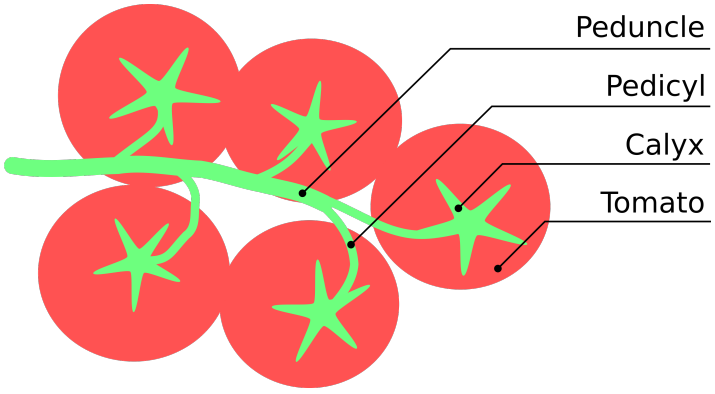


Figure 1-1: Naming conventions for vine tomato. The stem is made up of the peduncle, pedicel and calyx combined.

harvested manually by cutting the peduncle while supporting the truss as shown in Figure 1-2a. The fresh produce is transferred from the orchards or greenhouses in large trays, where it is packed according to the customers needs as shown in Figure 1-2b. This is done in bulk crates of several kilograms or in smaller quantities, such as 500gr packages[2]. In both cases trusses need to be added, removed or split by the staff, to meet a certain weight threshold while minimizing overpacking. This task may be combined with quality assessment and control, where small and damaged tomatoes are removed.

(a) Harvesting a ripe tomato truss in a greenhouse [3].

(b) A manned packing station in an industrial setting [4].

Figure 1-2: In contrast to tomatoes grown for processed consumption, the preparation of fresh vine tomatoes relies on manual labor.

1-1 Problem Statement

Mechanization and automation has tremendously increased crop production over the course of history. Further automation of harvesting and food processing by employing robotics may diminish the impact of labor shortages, reduce the financial risk for farmers and increase agricultural efficiency [5, 6]. One of the main challenges for developing agricultural automation is how to handle fragile and deformable organic objects without damaging the crop [7]. Such damage may be prevented by utilizing contact sensors or a detailed mechanical model of the object. However, contact sensors may not withstand the rough environment and high water contents of the organic materials. Furthermore, regular cleaning is necessary to meet strict hygiene regulations, but these sensors can make cleaning a cumbersome task. Finally, obtaining and simulating a mechanical model for deformable and fragile object is complex [8, 9]. Therefore, a solution is desired which may handle these fragile deformable objects while solely relying on geometric information and position-based control.

The goal of this thesis is to develop a novel geometry-based grasping method for vine tomato. The proposed grasping method utilizes a geometric model of the robotic hand and truss to determine an optimal grasping location on the peduncle. This method allows for grasping a truss without requiring contact between to robotic hand and fruit flesh. While the geometric features of the end effector may be determined in advance, the truss features need to be extracted by a vision system, because of the wide variety of shapes and sizes the crop

[2] A video of this process can be found here: https://youtu.be/XnqwZ_Y0I4A

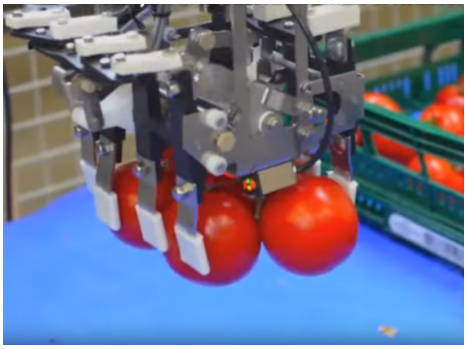
encompasses. To verify the geometry-based grasping approach for vine tomato, real world experiments are required since it is difficult to model the broad variation within the crop.

1-2 Background

While significant research effort has been put into the grasping and harvesting of tomato [10] only a few works mention vine tomato. This crop cannot be grasped in a similar manner since grasping an entire truss by a single fruit may easily damage the tomato, or cause it to detach from the truss. Therefore, key aspects of related prior works on vine tomato grasping, and tomato and stem detection are reviewed.

1-2-1 Vine Tomato Grasping

The PicknPack project developed and demonstrated a robotized production line to assess the quality and pack fresh and processed food products such as vine tomato [11]. The grasp action consists of several steps[3]. First, a pre-grasp is executed by performing a pinch grasp with the thumb of the custom designed end-effector. The truss is moved to a location where the end effector has sufficient space to pinch grasp the truss. Than a full grasp is performed, as shown in Figure 1-3a. The robot causes mild damage to the tomato skin as shown in Figure 1-3b. Ji et al., 2014 [12] designed a truss tomato harvesting robot for a greenhouse environment. After cutting, the truss is pinch grasped with two rubber plated fingers at the end of the peduncle. As a result the grasped truss hangs almost vertically downwards, making accurate placement impossible. Kondo et al., 2009 [13] use a similar approach for grasping vine tomato.



(a) A pinch based grasp on applied to the tomatoes.

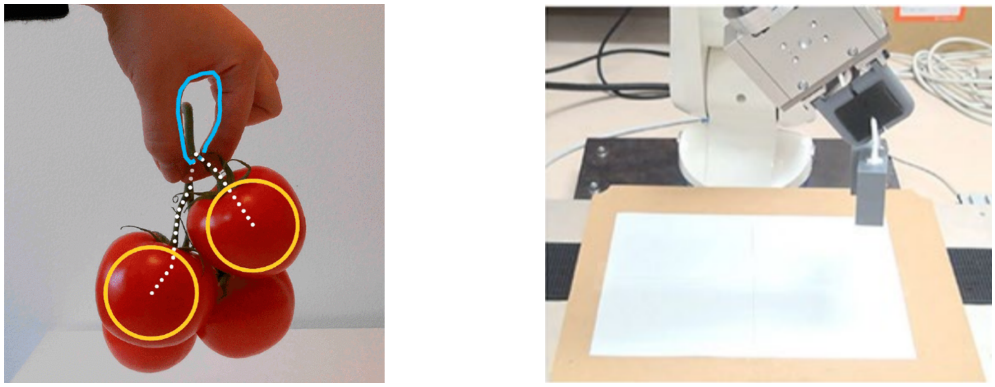
(b) Mild skin damage after picking and placing the truss.

Figure 1-3: The grasping method for vine tomato as presented by the PicknPack project [11].

Alternatively to these pinch grasp methods, caging can be applied. Caging bounds a target object's free path, without requiring tight contact between the object and end effector [14]. Benefits are low mechanical stress applied to the target object, robustness to minor control errors, and no need for force feedback and a mechanical object model. Varava et al., 2016 [15] determine sufficient caging conditions on rigid and partially deformable objects which exhibit double fork and neck features. For deformable objects, it is assumed that these features are

[3] A demonstration video can be found here: https://youtu.be/A4nk_1_oFWk

preserved under deformations. An example of caging applied to a tomato truss is shown in Figure 1-4b. Caging cannot determine an object's pose uniquely, and may only be used to manipulate objects roughly. Maeda et al., 2012 [16] introduce caging based grasping which adopts the advantages of both grasping and caging approaches, by constraining objects using only simple position control. Effectively, this means that the rigid parts of the end effector form a cage which the object cannot escape, while the soft parts deform and apply a reaction force to the object. Egawa et al., 2015 [17] and Kim et al., 2019 [18] analytically compute the required gripper distance to cage various rigid and deformable objects based on a geometric model. Among others, it is demonstrated on an object consisting of two connected cubes via a rope, similar to a tomato truss as shown in Figure 1-4b.



(a) Caging a tomato truss [15]. **(b)** Caging a string and rigid cubes [18].

Figure 1-4: Caging may be used to lift vine tomato by utilizing its neck feature - a thin part with thicker ends at both sides.

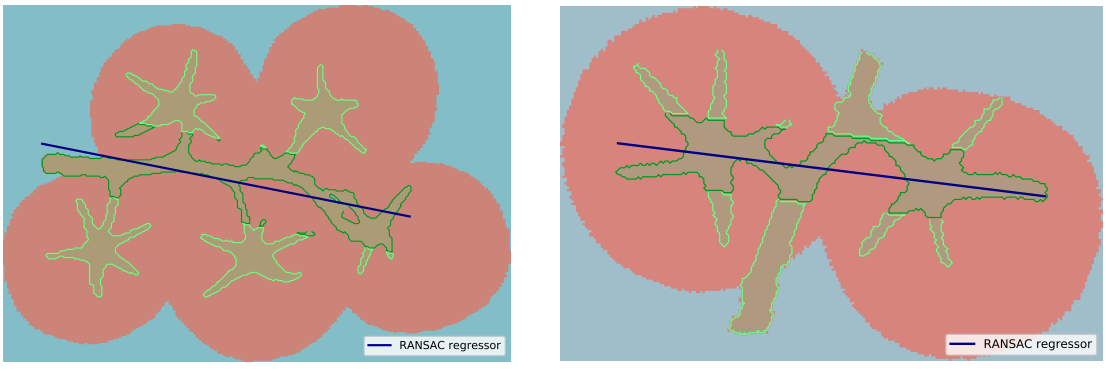
1-2-2 Tomato and Stem Detection

A wide range of computer vision techniques have been developed for tomato detection. A popular approach is color-based segmentation using various color descriptions. Feng et al., 2018 [19] make use of the difference between red and green color components (R-G) to select an area where a truss is present. Lili et al., 2017 [20] apply Otsu's method to the normalized R-G component to differentiate between background and tomato. Other color models than RGB have been applied, with desired characteristics for tomato detection. Feng et al., 2015 [21] utilize the hue, saturation and intensity (HSI) color space to identify the ripe fruit using predetermined threshold values. Malik et al., 2018 [22] apply a combination of adaptive threshold segmentation and predetermined threshold in the hue, saturation and value (HSV) color space. Connected tomatoes are separated using a watershed algorithm. Alternatively, multiple color spaces can be combined. Zhao et al., 2016 [23] fuse the a* color component from the L*a*b* color space with the I component from the luminance, in-phase and quadrature-phase (YIQ) color space. The target tomato is separated from the background with adaptive threshold segmentation.

Instead of solely relying on color information to detect tomatoes, geometric features can be added. Zhoa et al., 2016 [24] recognize ripe tomatoes by combining color analysis in the I component of the YIQ color space, with Adaboost classification based on Haar-like features.

Lin et al., 2020 [25] detect sub-fragments of the fruits by matching contour information with a predefined shape descriptor. Sub-fragments are aggregated using a novel probabilistic Hough transform. Fruit candidates are verified by a support vector machine trained on HSV values and texture features. Yuan et al., 2020 [26] propose a deep learning approach and provide a comparison between various Single Shot multi-box Detector networks. Liu et al., 2020 [27] deal with complex environment conditions, such as occlusion, overlap and illumination variation by developing a model based on YOLOv3 with two modifications. Their method replaces the traditional rectangular bounding boxes with circular bounding boxes which matches the shape of a tomato better. Furthermore, incorporation of direct connection between different layers allow for feature reuse and help to learn more accurate models.

Only few works aim to identify the truss stem. Ji et al., 2014 [12] detect a picking point on the stem by using the white tomato-clips. These are placed on the stem to avoid bending. Kondo et al., 2009 [13] utilize the HSI color space to detect the connection of a truss to the tomato plant. Yoshida et al., 2019 [28] train a support vector machine on RGB data to separate tomatoes from the background. Individual clusters and a cutting point on the peduncle are identified based on the geometry of the point cloud. Benavides et al., 2020 [29] combine edge detection with color-based segmentation to identify both the fruit and stem of the truss. The PicknPack project developed a vision system to detect vine tomatoes placed in a single non-overlapping layer in blue harvest crates [30]. Images are labeled in order to learn several parameters, such as tomato and stem color. Tomato trusses are segmented into stem and tomato flesh based on principle component analysis (PCA). Individual tomatoes are detected by fitting ellipses with a Hough transform. The peduncle is identified using a Random sample consensus (RANSAC) regressor based on the assumption that the peduncle is the longest pixel area present in the stem segment. This assumption does not always hold, causing the peduncle classification to fail as shown in Figure 1-5.



(a) Successful identifications of the peduncle. **(b)** Misclassification of the peduncle.

Figure 1-5: a RANSAC regressor is applied to identify the truss peduncle. The identified peduncle is encircled with dark green, the other stem parts with light green. On the large truss it successfully identifies the peduncle. The assumption that the peduncle makes up the larges line in the image fails for the smaller truss.

1-3 Conclusion

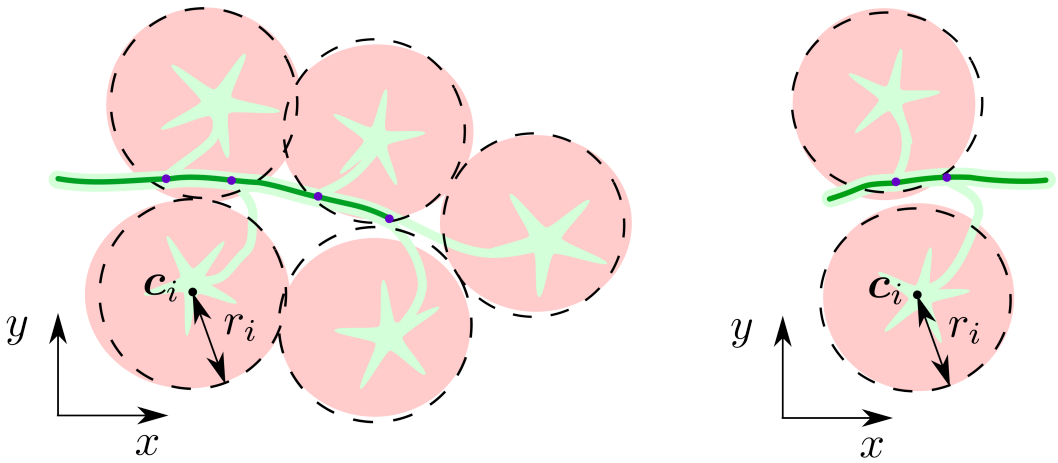
Contrary to the research effort put in tomato harvesting and detection, not much literature exists on the grasping and identification of vine tomato. The three methods applied to vine tomato utilize a pinch based grasp, two at the end of the peduncle making the pose of the truss uncontrollable [12, 13], and one at the skin of the tomato resulting in damage done to the fruit [11]. Where tomato detection has reached impressive results in complicated environment conditions, stem detection is mentioned only a few times. Only a single work identified the truss peduncle, but the approach fails for trusses where the peduncle does not make up the largest area of the stem [30].

The main contributions of this thesis are (i) the application of a geometry-based grasping method to vine tomato, (ii) the development of a vision system which identifies relevant features of vine tomato, and (iii) the validation of the method using real world robotic experiments. After this introduction Chapter 2 proposes the geometry-based grasping method for vine tomato. Chapter 3 describes the computer vision pipeline with a novel peduncle detection method which extracts the required features for the geometric model. In Chapter 4 the robotic experiments are described, and the obtained results are presented and discussed. Finally, Chapter 5 concludes this thesis.

Chapter 2

Geometry-Based Grasping Method

A geometry-based grasping method for vine tomato is proposed. The method uses a geometric model of the truss and end effector to determine a grasp location and orientation (pose). The grasp needs to satisfy the caging-based grasping conditions, for which the constraints will be derived. These conditions are satisfied at various locations on the peduncle, from these options an optimal grasp pose will be determined. It is assumed that (i) the truss is placed on a horizontal surface, (ii) sufficient space is available above the truss for the manipulator to approach the target object from above, (iii) the target truss is separated from other trusses, and (iv) the peduncle lies upwards.



(a) A truss containing five tomatoes.

(b) A truss containing two tomatoes.

Figure 2-1: An vine tomato overlapped with its geometric model. The tomatoes are represented by the dashed circles (○), the peduncle by the dark green line (—), and the junctions by the purple dots (●).

2-1 Geometric Models

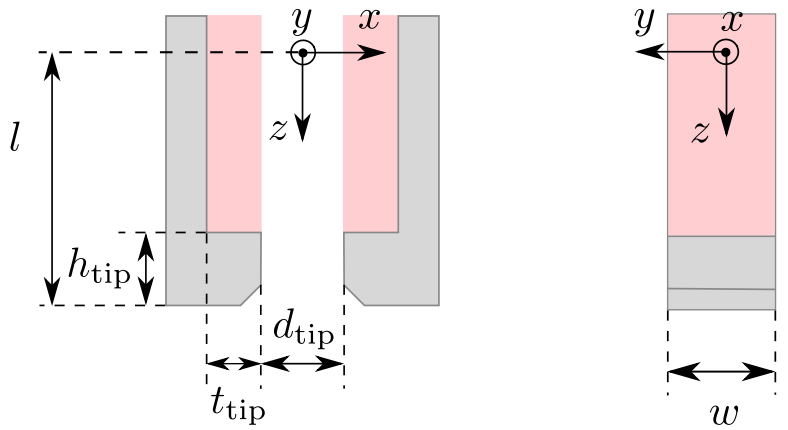
A geometric model of the tomato truss and end effector will be introduced. A two dimensional truss model is used since it can describe most features of a truss, whereas incorporating three dimensional information requires accurate depth data, which is difficult to obtain and more complex to process.

2-1-1 Tomato Truss

The geometric truss model exploits knowledge on the crop's morphology. Models for two different trusses are shown in Figure 2-1. A truss is made up of tomatoes and a stem. The tomato i is modeled as a circle with radius r_i and center $\boldsymbol{c}_i \in \mathbb{R}^2$. The stem consist of a peduncle, calyces and pedicels. These pedicels split of the peduncle at various locations, which will be referred to as junctions. Generally, a pedicel is connected to a tomato via the calyx. However, this is not necessarily the case since a tomato may have been detached from the truss. The peduncle is modeled by a finite set of branches which either connect the end of the peduncle to a junction, or connect two junctions.

2-1-2 End effector

The end effector model is shown in Figure 2-2. It is modeled as a parallel gripper of length l, and width w. Fingertips of height h_{tip} and thickness t_{tip} are attached to the parallel gripper to create L-shaped fingers. The distance between the finger tips d_{tip} can be controlled. The inside of these fingers are covered with soft parts which give better control of the target object's pose. This is required to meet the caging-based grasping conditions as discussed next.



(a) Side view **(b)** front view

Figure 2-2: Geometric model of the end effector, shown from two different perspectives. The red area marks the deformable material.

2-2 Grasp Constraints

The geometric models are used to derive concrete constraints to meet the caging-based grasping conditions. A caged object can move freely in the closed region unless the region is a single point, which is called form closure. Maeda et al., 2012 [16] introduce caging based grasping which uses deformable parts on the gripper to apply a reaction force to the object. The following two constraints need to be satisfied for a caging-based grasp:

1. **Rigid-part caging condition**: the target object is in a complete imprisonment of the rigid parts of the end effector.

2. **Soft-part caging condition**: assuming that the soft parts of the end effector are rigid, the closed region for caging in the configuration space of the object becomes empty.

The former condition implies that the object is inescapable from the end effector, while the latter condition implies that the soft parts deform, and thus apply a reaction force to the target object. Both conditions can be tested geometrically.

The rigid-part caging condition can be verified by noting that the truss stem exhibits various double fork features. These features, as defined by Varava et al. [15], can be described by a curve whose ends spread sufficiently far to prevent the loop formed by the end effector from moving arbitrarily far away from the object. In the case of vine tomato, pedicels are connected at multiple junctions to the peduncle. Therefore, the rigid-part caging condition is met when the end effector encloses the branches encapsulated by these junctions as shown in Figure 2-3. It is assumed that these double fork features are preserved under deformations of the stem. The peduncle is enclosed when the fingertip distance is smaller than the peduncle diameter. Since the peduncle diameter is not included in the geometric model, the end effector will simply be completely closed. The geometry of the tomatoes is not used, thus a cage may be applied to a truss even when tomatoes are missing. The soft-part deformation condition is satisfied when the diameter of the peduncle is larger than the fingertip distance, this requirement is automatically satisfied when the end effector is fully closed.

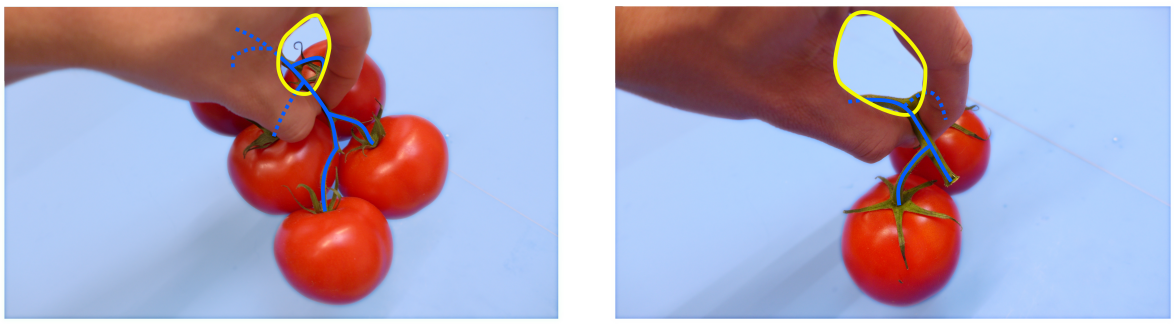


Figure 2-3: A human hand applies a cage to a tomato truss utilizing the double fork features of the stem. The ends of the blue curve are spread sufficiently far to prevent the truss from moving arbitrarily far from the yellow loop formed by the human hand.

2-3 Grasp Pose

To determine a target grasp location on the branches encapsulated by junctions two additional conditions are introduced:

- **Space condition**: the target grasp location lies at least a length l_{thresh} away from the junctions encapsulating the branch.
- **Balance condition**: The target grasp location lies as close to the center of mass as possible.

The space condition is introduced such that there is sufficient space for the end effector on the target grasp location. The distance threshold l_{thresh} is taken as dependent on the end effector finger width as:

$$l_{\text{thresh}} = c \cdot w \tag{2-1}$$

Where $c \geq 1$ is a safety factor. Note that generally a value larger than one is chosen, since in realisty the actual end effector pose will differ from the target pose due to control errors. The balance condition is introduced to prevent undesired tiling of the truss. Two assumptions are made to estimate the center of mass (i) the stem mass is negligible compared to the tomato mass, (ii) the tomato mass scales with the volume of a sphere with an identical radius [31]. The center of mass $\boldsymbol{c}_{\text{com}}$ may be determined as follows:

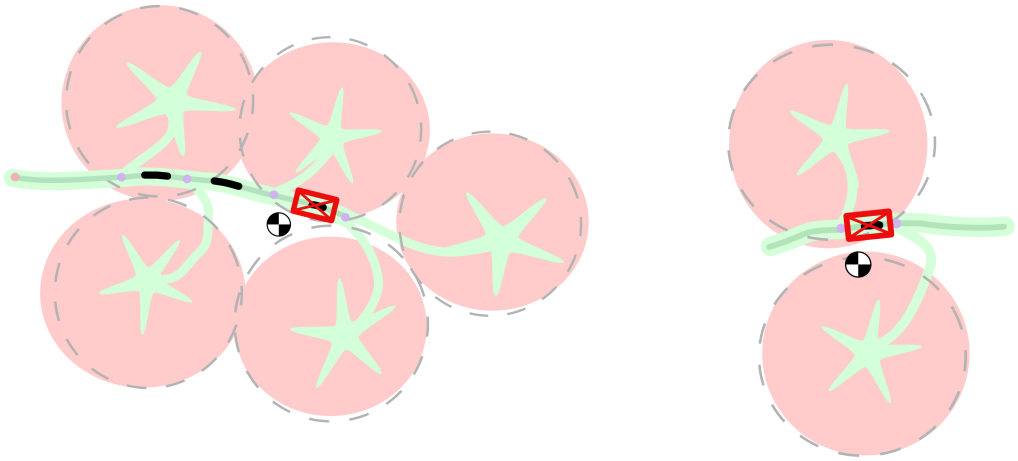
$$\boldsymbol{c}_{\text{com}} = \frac{\sum_{i=1}^{N_{tom}} r_i^3 \boldsymbol{c}_i}{\sum_{i=1}^{N_{tom}} r_i^3} \tag{2-2}$$

The selected grasping point is the candidate location which lies closest to the center of mass.

The end effector is aligned with the yaw, or the rotation around the z-axis, of the peduncle. This orientation is determined by taking the arctangent of the difference between junctions encapsulating the target branch. The determined target grasp pose for the previously discussed trusses are shown in Figure 2-4. This two dimensional grasp pose is transformed to 3D by adding the grasp height, which is derived from the peduncle height and end effector dimensions. To grasp a truss, the following routine will be executed. First, the manipulator moves to a pre-grasp location which lies vertically above the target grasp location. The end effector is moved downwards to the grasp pose, where the end effector is closed. Once a caging-based grasp is obtained the end effector is moved back to the pre-grasp location, from where the robot may be commanded to place the object.

2-4 Summery and Concluding Remarks

A geometry-based grasping method for vine tomato is proposed. The method utilizes the double fork features of the truss stem, assuming that these features are preserved under deformations. It may be applied to trusses of various shapes and sizes, even when tomatoes are missing. A space condition is used to determine whether a target branch is sufficiently



(a) A truss containing five tomatoes. **(b)** A truss containing two tomatoes.

Figure 2-4: The geometric model overlapped with the determined grasping location. The center of mass is shown as a crossed circle (◓), the possible grasping locations are the black lines (▬), and the target grasping location is shown by the red box (✉).

long for the end effector fingers. However, it does not take into account whether there is sufficient space during the approach and closing of the end effector. Furthermore, the space present around the peduncle is not checked, it may lie in between two tomatoes making it unreachable. The balance condition is introduced to prevent undesired tilting of the truss. To determine the center of mass the tomatoes are modeled as a circle, whose mass is proportional to the volume of a sphere with an identical radius. This simplification may not work well for all tomato cultivars, resulting in an inaccurate center of mass prediction. Li et al., 2011 [32] propose a three-dimensional geometric tomato model which uses 5 dimensions to calculate various features of the tomato such as mass. Nyalala et al., 2019 [31] estimate the mass and volume of a tomato using regression prediction models based on six features. To apply the grasping method several features of the truss need to be identified. This is done with a computer vision pipeline, as discussed in the next chapter.

Chapter 3

Computer Vision Pipeline

A computer vision pipeline is developed to extract geometric tomato and peduncle features from RGB images. Despite the rise in popularity of deep learning in the field of computer vision over the last years [33], a more traditional approach will be taken. While deep learning performs well at versatile problems containing many possible outcomes such as object detection, the problem at hand is more confined. Features of a single object type are extracted in a well controlled environment. Moreover, a deep learning approach is data hungry and no suitable dataset is readily available. A traditional approach based on handcrafted features, provides more insights into the problem and allows for tuning with easy to interpret parameters. The computer vision pipeline consists of several layers. After the images are taken, they are segmented, filtered, and cropped. On these cropped images, the tomato and peduncle features are extracted. Finally the pipeline is tested, the results are presented and discussed.

3-1 Data Acquisition

An industrial setting is assumed, where trusses are placed in a single non-overlapping layer in blue harvest crates. The images must comply with the following requirements:

- The image contains a single truss
- The stem lies upwards
- The truss is non-occluded
- The image is taken from top view
- The background is blue

In addition to the RGB data, an approximation of the number of pixels per millimeter is determined based on the camera intrinsics and depth information. To evaluate the computer vision system, twelve trusses have been photographed and labeled at seven different poses each resulting in a total of 84 images. Some images are shown in Figure 3-1. A more detailed description, and an explanation of the labeling process can be found in Appendix A.

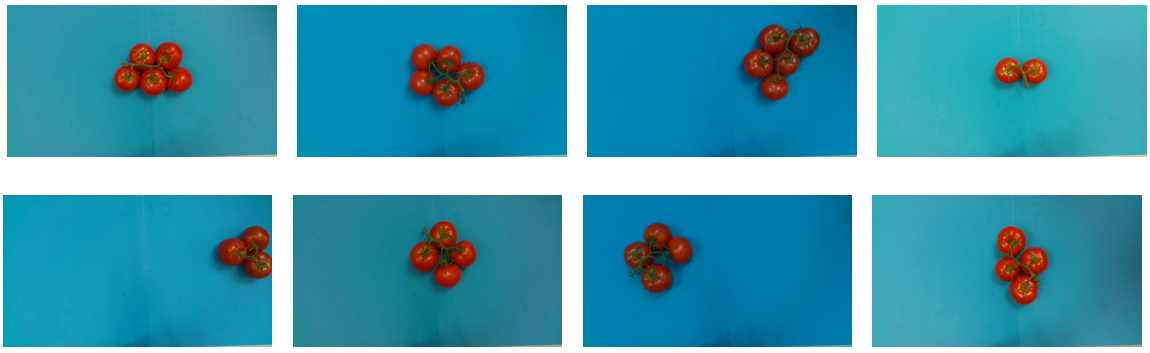


Figure 3-1: Several examples from the dataset containing 84 labeled images of real tomato trusses.

3-2 Color Space

The captured images are represented in the red, green and blue (RGB) color space. The combination of colors are modeled as how red, green and blue light are added together considering a black base color. It is undesired to use the RGB representation directly, since the separate color components are strongly correlated and share luminosity information together. This makes color segmentation independent of the light conditions difficult. Furthermore, humans do not perceive color as a combination of three primary colors, making it a non-intuitive color space for image processing [34]. Many alternative color spaces have been developed according to more subjective entities related to luminosity, hue, and saturation. The two color components which will be used for this pipeline are discussed below. Information about some other color spaces and a comparison can be found in Appendix B.

The hue color component is an angular quantity, which starts at red (0°), passes through green (120°) and blue (240°), and wraps back to red (360°). It is based on the maximum M, minimum m and chrome C components, which are defined as:

$$\begin{aligned} M &= \max(R, G, B) \\ m &= \min(R, G, B) \\ C &= \text{range}(R, G, B) = M - m \end{aligned} \tag{3-1}$$

Where R, G, and B are the pixel color values. These are used to define hue H piecewise as follows:

$$H = 60° \cdot \begin{cases} \text{undefined}, & \text{if } C = 0 \\ \frac{G-B}{C} \bmod 6, & \text{if } M = R \\ \frac{B-R}{C} + 2, & \text{if } M = G \\ \frac{R-G}{C} + 4, & \text{if } M = B \end{cases} \tag{3-2}$$

For all gray tones ($R = G = B$) the hue value is undefined. Figure 3-2a shows the hue color component of an image, Figure 3-2c provides the raw values. Based on the pixel hue it is possible to distinguish between the truss and blue background independently of brightness.

However, the hue values of the tomatoes and stem are similar. To distinguish between these objects, an additional color component is used.

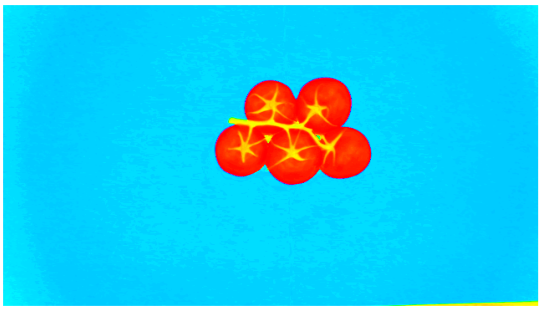
The L*a*b* color space was developed, such that a numerical change corresponds roughly to the visually perceived change of the color. The a* color component distinguishes between green (low value) and red (high value) colors. First the X, Y and Z values are computed from the RGB image using a linear transformation. It is assumed that the RGB values are provided in the sRGB standard, defined relative to a D65 reference white. The constants used for the transformation are specified in the CIE international standard [35]. The obtained XYZ values are used to construct the a^* color component as follows:

$$a^* = 500\left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right) \tag{3-3}$$

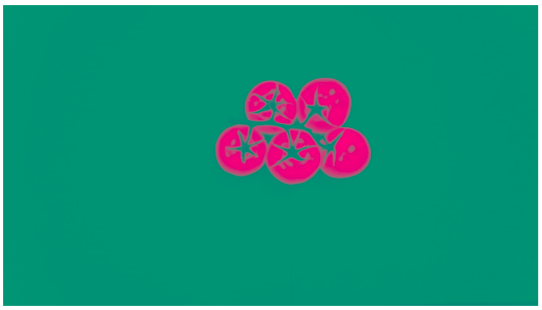
With

$$f(t) = \begin{cases} \sqrt[3]{t} & \text{if } t > \delta^3 \\ \frac{t}{3\delta^2} + \frac{4}{29} & \text{otherwise} \end{cases} \tag{3-4}$$

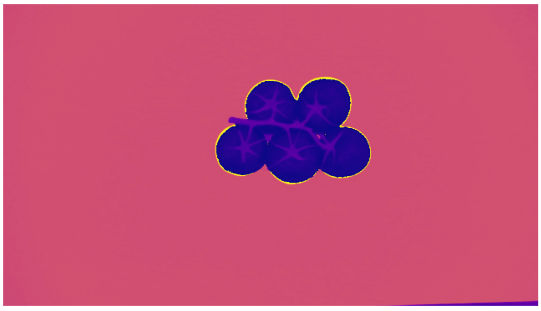
Where $\delta = \frac{6}{29}$ and X_n, Y_n and Z_n are the tristimulus values corresponding to the D65 reference white, as defined in [36]. In Figure 3-2b the a* color component of an image is visualized. Figure 3-2d provides the raw values.



(a) The hue color component, transformed back to RGB using maximum saturation and medium brightness.



(b) The a* color component, transformed back to RGB using medium b* and brightness.



(c) The raw hue values, scaled down by a factor two such that it fits the [0,255] range.

(d) The raw a* values, translated by +125 such that it fits the [0,255] range.

Figure 3-2: The hue and a* values of an image.

3-3 Image Segmentation

Image segmentation aims to change the representation of an image such that it is easier to analyze. A label is assigned to all pixels in a way that pixels with the same label share certain characteristics. In this particular case, the labels used are background, stem, and tomato. In literature, various segmentation methods have been proposed based on color, texture, shape or a combination of these features [37]. Because of the distinct color characteristics corresponding with each label, color based segmentation is applied. By utilizing the previously discussed color components in combination with an adaptive threshold segmentation algorithm, the vision pipeline should be sufficiently insensitive to light conditions and truss variations.

3-3-1 Adaptive Threshold Segmentation

The key part of segmentation is determining an appropriate threshold. This is done using a clustering algorithm. A popular method is color based k-means clustering, where the n observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_n$ are partitioned into one of the k sets $\boldsymbol{S} = \{S_1, S_2, \dots S_k\}$ to minimize the sum of squared euclidean distances between observation and cluster mean $\boldsymbol{\mu}_j$. The objective is formulated as:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||\boldsymbol{x} - \boldsymbol{\mu}_i||^2 \tag{3-5}$$

This is equivalent to minimizing the pairwise squared deviations of points in the same cluster. A major computational advantage is that this distance can be minimized without explicitly using the pairwise distances between all the data points. It is often solved using an iterative refinement technique commonly referred to as the k-means algorithm or Lloyd's algorithm.

3-3-2 Data Preparation

The hue and a* matrices are downsampled by a factor two, the result is stored into n-sized column vectors $\boldsymbol{h}$ and $\boldsymbol{a}^*$. K-means clustering works in euclidean space and cannot take into account the periodic nature of angular data. Therefore the hue angles are transformed to points on a circle with radius r. The a* color component is normalized into $\bar{\boldsymbol{a}}^*$ such that it lies on the [-1, 1] interval:

$$\bar{\boldsymbol{a}}^* = 2\left(\frac{\boldsymbol{a}^* - \min(\boldsymbol{a}^*)}{\max(\boldsymbol{a}^*) - \min(\boldsymbol{a}^*)}\right) - 1 \tag{3-6}$$

The three-dimensional observations are constructed as:

$$\boldsymbol{x}_i = \begin{bmatrix} r\cos \boldsymbol{h}_i \\ r\sin \boldsymbol{h}_i \\ \bar{\boldsymbol{a}}_i^* \end{bmatrix}, \quad \forall i = 1, 2, \dots, n \tag{3-7}$$

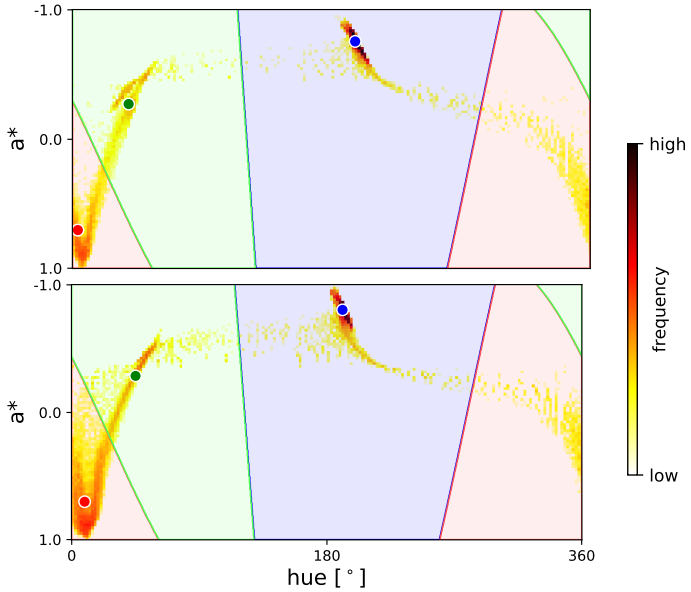
These observations lie on a cylinder with a fixed length of two and radius r. Increasing the radius affects the impact of the hue color component on the clustering relative to a*. Decreasing r generally makes the algorithm more sensitive to labeling a pixel as peduncle, resulting in more true and false negatives. A good balance is found for $r = 1.5$.

3-3-3 Initialization and Stopping Criteria

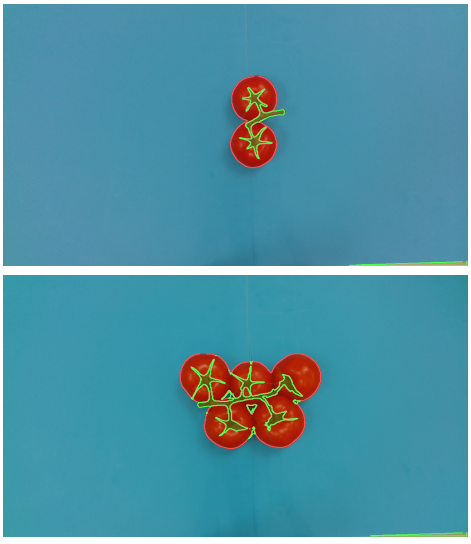
The three centers are initialized near a red, green, and blue color corresponding to the tomato, peduncle, and background segments. K-mean clustering is stopped either when the cluster centers move by less than a specified accuracy ϵ, or when a maximum amount of iterations $i_{\max}$ is reached. These parameters affect the computational cost and accuracy of the algorithm. By choosing mild termination criteria the algorithm will be fast, but may not converge, resulting in low accuracy. This can be improved by making the criteria more strict, at the cost of computational time. Based on this trade-off the stopping criteria are set to $\epsilon = 0.01$ and $i_{\max} = 20$. The determined cluster centers are used to segment the original image. The three segments are labeled based on the assumption that the tomato cluster center hue value is red and the background hue value is blue. The stem label is assigned to the remaining cluster.

3-3-4 Discussion

The two-dimensional histograms in Figure 3-3a visualize the color values and cluster centers per image. The background segment is separable from the truss, however the tomatoes and peduncle have some overlap in this two-dimensional space. Most information for assigning a pixel is in the hue value. The a* color component mainly affects the classification boundary between the stem and tomato segment. The obtained segments are shown in Figure 3-3b. Some parts on the tomato edges are misclassified as stem, but these can easily be removed using morphological operations which will be discussed next. Background pixels in between



(a) Histogram based on the a* and hue values of each pixel in the image. The frequency scales logarithmically. The dots represent the cluster centers and the colored areas mark the label assigned to the sample.



(b) The original image overlaid with the determined class contours.

Figure 3-3: K-means clustering applied to the hue and a* color component. Red color corresponds to the tomato class, green color to the stem, and the blue color to the background.

the tomatoes are occasionally misclassified as tomato or stem. These pixels are very dark in the original images and do not contain sufficient color information to make a reliable prediction. This might negatively impact the detection of tomatoes and peduncle later in the pipeline.

3-4 Filtering

Pixel blobs are filtered from the segmented image such that it will be easier to analyze. Small areas in the background are misclassified as tomato or stem. To remove these blobs, first the tomato and stem segment are combined into a new truss segment. Only the contour with the largest area is kept, all others are assigned to the background segment. Similarly, the tomato edges which are misclassified as stem are assigned to the tomato segment. All contours in the stem segment but the one with the largest area are labeled as tomato. Finally, morphological opening and closing is applied to the stem segment with a circular kernel diameter d_{kernel} of 3 pixels to remove all thin edges. The result of these morphological operations are shown in Figure 3-4.

(a) The original segmented image. **(b)** The filtered segmented image.

Figure 3-4: A comparison between the segmented image before and after filtering. In the original image the lower right corner is misclassified as peduncle. Furthermore, some parts in between and on the edges of the tomatoes are misclassified as peduncle. This is fixed by the applied filters.

3-5 Cropping

To speed up and improve the performance of the tomato detection, the image is cropped around the truss using a rotated rectangle. This procedure consists of four consecutive steps. (i) The angle α between the truss and the horizontal axis of the image is determined, (ii) the image is rotated by $-\alpha$ such that the truss is aligned with the horizontal axis, (iii) a bounding box is determined, (iv) the image is cropped using this bounding box.

The central moments μ_{pq} of order $p+q=2$ can be used to determine the orientation of the grayscale image $I(x,y)$. The covariance matrix of this image is given by:

$$\text{cov}\left[I(x,y)\right] = \begin{pmatrix} \mu'_{20} & \mu'_{11} \\ \mu'_{11} & \mu'_{02} \end{pmatrix} \tag{3-8}$$

Since the eigenvectors of this matrix correspond to the minor and major axes of the image, the orientation can be determined by finding the angle of the eigenvector with the largest eigenvalue. It can be shown that this angle is equal to:

$$\alpha = \begin{cases} \frac{1}{2}\arctan\left(\frac{2\mu'_{11}}{\mu'_{20}-\mu'_{02}}\right), & \text{if } \mu'_{20}-\mu'_{02} \neq 0 \\ -\pi/4, & \text{if } \mu_{11} < 0 \\ \pi/4, & \text{else} \end{cases} \tag{3-9}$$

The cropping is done using the scikit-image toolbox[1]. The truss orientation is determined based on the stem segment. The rotation found is applied around the center pixel such that the truss lies horizontally. The bounding box is computed such that it surrounds all tomato and stem pixels.

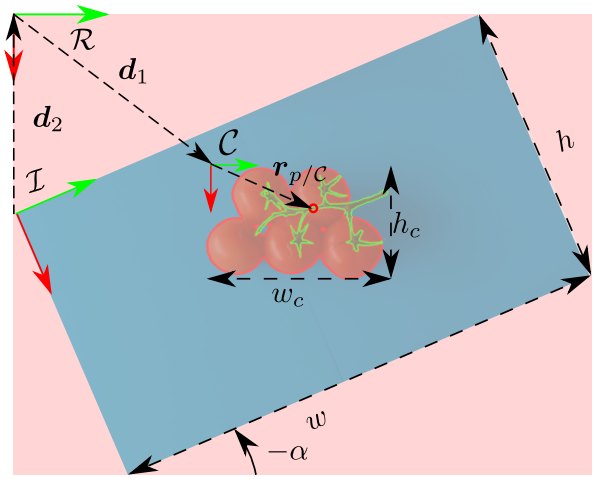


Figure 3-5: Cropping applied to an image.

Coordinates found in the cropped image need to be transformed back into the original image. This requires a 2D coordinate transformation as shown in Figure 3-5. The image frame $\mathcal{I}$ is attached to the upper left corner of the original picture, the rotated frame $\mathcal{R}$ is attached to the upper left corner of the rotated image, and finally the cropped frame $\mathcal{C}$ is attached to the upper left corner of the cropped image. We denote the position of point p with respect to the reference frame $\mathcal{C}$ as $\boldsymbol{r}_{p/\mathcal{C}}$. This frame may be expressed in the rotated and image frame as follows:

$$\begin{aligned} \boldsymbol{r}_{p/\mathcal{R}} &= \boldsymbol{r}_{p/\mathcal{C}} + \boldsymbol{d}_1 \\ \boldsymbol{r}_{p/\mathcal{I}} &= {}^{\mathcal{R}}\mathrm{R}_{\mathcal{I}}\left(\boldsymbol{r}_{p/\mathcal{C}} + \boldsymbol{d}_1 + \boldsymbol{d}_2\right) \end{aligned} \tag{3-10}$$

Where:

- $\boldsymbol{d}_1$: is the displacement vector from the origin of $\mathcal{R}$ to the origin of $\mathcal{C}$
- $\boldsymbol{d}_2$: is the displacement vector from the origin of $\mathcal{I}$ to the origin of $\mathcal{R}$

[1] https://scikit-image.org

- ${}^{\mathcal{R}}\mathrm{R}_{\mathcal{I}}$: is the rotation matrix which expresses the orientation of $\mathcal{I}$ with respect to $\mathcal{R}$

The first displacement depends on the determined bounding box. The second displacement can be determined using trigonometry as a function of the rotation angle and original image dimensions. Two cases dependent on the direction of rotation need to be distinguished:

$$\mathbf{d}_2 = \begin{cases} \begin{bmatrix} -w \sin \alpha \\ 0 \end{bmatrix} & \alpha < 0 \\ \begin{bmatrix} 0 \\ h \sin \alpha \end{bmatrix} & \alpha > 0 \end{cases} \tag{3-11}$$

Finally, the two-dimensional rotation matrix is defined as:

$${}^{\mathcal{R}}\mathrm{R}_{\mathcal{I}} = \begin{bmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{bmatrix} \tag{3-12}$$

Now feature extraction can be performed on the cropped images, and the result can be transformed to the original image.

3-6 Tomato Detection

The tomatoes are modeled as simple circles, whose centers and radii need to be determined. This is done by applying the Circle Hough Transform (CHT). A tomato edge may be described as:

$$(x - x_{center})^2 + (y - y_{center})^2 = r^2 \tag{3-13}$$

Where x and y are the edge coordinates. The three dimensional parameter space consists of center (x_{center}, y_{center}) and radius r. Since using a 3D accumulator space for the hough transform would be highly ineffective, the Hough Gradient Method is used which splits the circle fitting process into two stages as explained by Yuen et al., 1990 [38]:

1. The circle center lies where the normals of many edge points intersect. The votes along the normal of each point are accumulated into a two-dimensional accumulator array.
2. The radius is determined based on the distance of each point from a candidate center. A radius histogram is used to identify the radius of the circles.

This method is implemented in the cv2 library[2]. The tomato segment is used as source image. The circle edge detection is implemented via the Canny algorithm, which requires a gradient to be present in the image. This is created by blurring the segment with a Gaussian kernel of size d_{blur} set to 3 and standard deviation σ_{blur} set to 0.8. The choice of these parameters does not significantly impact the results, as long they are in a reasonable range. Then the Hough Gradient Method is applied using the following parameters:

- The minimum circle radius $r_{\min}$ is set to 30mm.

[2] https://opencv.org/

- The maximum circle radius $r_{\max}$ is set to 40mm.
- The minimum distance between the circle centers $d_{\min}$ is set to the minimum tomato diameter.
- The gradient threshold c_1 for accepting a pixel as an edge is set to 20.
- The inverse ratio between accumulator resolution and image resolution c_{dp} is set to 4 to decrease the computational burden while maintaining a sufficient accuracy.
- The accumulator threshold c_2 for the circle centers at the detection stage is set to 80.

False positives are removed by discarding all predicted circles which do not overlap the tomato segment by at least a factor c_{fill}. By taking $c_{\text{fill}} = 0.5$ at least half of the circle has to be filled with pixels of the tomato segment. The result on two images are shown in Figure 3-6.

(a) Good fit **(b)** Rough fit

Figure 3-6: Result of the CHT after filtering based on overlap. The identified tomatoes are marked by the dashed circles (), and the predicted center of mass is marked by the crossed circle ().

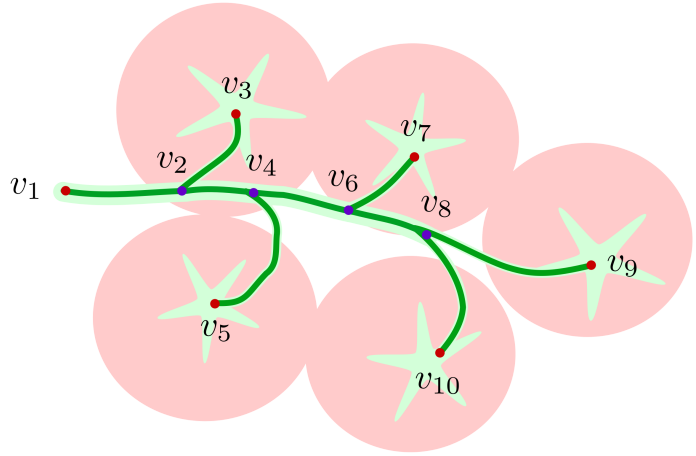
3-7 Peduncle Detection

Previous work aimed to identify the peduncle by fitting a line on the largest green pixel blob present in the image [30]. However, this method ignores the local curvature of the stem and fails for trusses where the peduncle does not make up the largest pixel area in the image. Therefore, a graph-based method is proposed which identifies the peduncle using length and curvature.

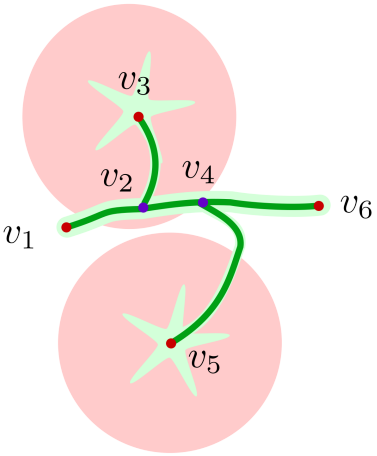
3-7-1 Graph Theory

The skeleton of the stem is analyzed using a graph search method. The stem is described as a weighted graph, which is an ordered triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Psi)$, where:

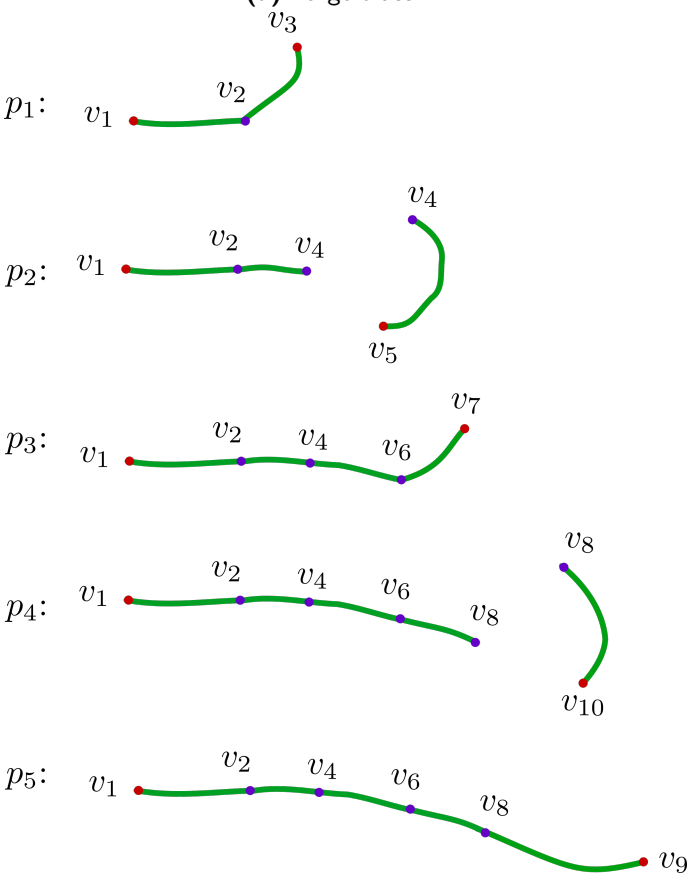
- $\mathcal{V} \subseteq \{\{x, y\} | (x, y) \in \mathbb{R}^2\}$ is a set of vertices with a two-dimensional location.
- $\mathcal{E} \subseteq \{\{u, v\} | (u, v) \in V^2\}$ is a set of edges connecting the vertices



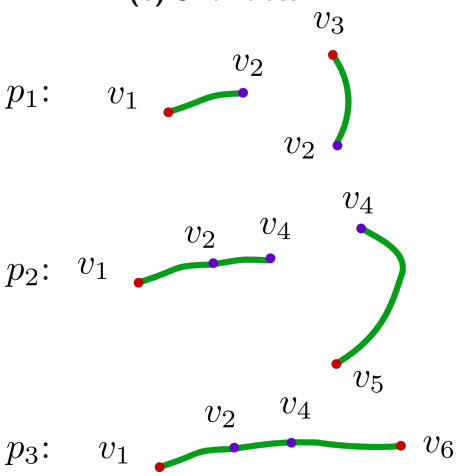
(a) Large truss.



(b) Small truss.



(c) Obtained paths 2 and 4 have been split, since the last edges exceed the orientation threshold. The largest path with limited curvature is from v_1 to v_9.



(d) Obtained paths 1 and 2 have been split, since the last edges exceed the orientation threshold. The largest path with limited curvature is from v_1 to v_6.

Figure 3-7: The upper row shows the graph obtained from the peduncle and pedicels of two different trusses. The second row shows the corresponding paths searched with origin v_1. Edges are marked by the dark green lines (—), junctions by the purple dots (●), and tails by the red dots (●).

- $\Psi(e) = (u, v)$ is an incidence function which associates each edge e with an unordered pair of vertices (u, v).

The different stem segments are represented as edges, interconnected via vertices. Each edge e is associated with a real number $w(e)$ called the weight. This weight represents the length measured along the edge. Let $d(v)$ denote the degree of a vertex, which is the number of edges incident with v. A vertex is either called a tail when $d(v) = 1$, or a junction when $d(v) = 3$. Two trusses of vine tomato, together with their graph are shown in Figure 3-7a and 3-7b.

A walk in the graph is a finite non-null sequence of alternating vertices and edges $H = v_0e_1v_1e_2v_2 \dots e_kv_k$ such that the ends of e_i are v_{i-1} and v_i for $1 \leq i \leq k$. H is a walk from origin v_0 to terminus v_k of length k. A walk is called a path P if the edges and vertices are distinct. The weight of such a path is defined as the sum of the weights on its edges. The orientation of an edge e is computed by taking the arctangent of the difference between the source and destination vertex of the edge. The angle of an entire path is computed using the origin and terminus vertex of the path.

3-7-2 Peduncle Search

It is assumed that the peduncle is the longest line on the stem with a limited curvature. The curvature between an edge and a path is approximated by the difference between their orientation. The peduncle is identified from the graph by searching for the path with the largest weight, where a path is split if the curvature exceeds a threshold. An path is generated staring from an arbitrary tail vertex towards another tail vertex. When the difference between the orientation of a newly observed edge and traversed path exceeds threshold α_{thresh}, a new path is started. The longest seen path is stored and compared to the newly generated path. If the length of the longest path is exceeded by the new path, it is updated. Once this search is performed starting from all tail vertices, the resulting longest seen path is the peduncle. An example of the search path when starting from key vertex v_1 is shown in Figure 3-7c and 3-7d. The search path is split several times because the curvature threshold is exceeded.

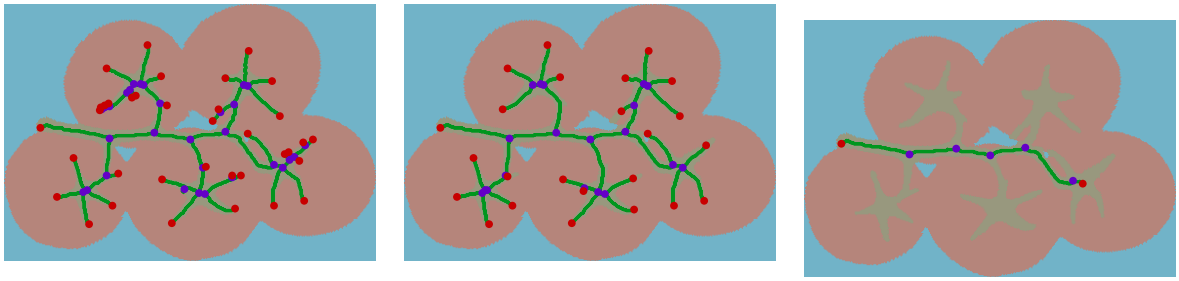
The algorithm searches for a path between a tail vertex to all other tail vertices in the graph, and repeats this for all tail vertices. This means that computational complexity grows rapidly with graph size. Therefore a path threshold is used to stop the algorithm from creating paths with a length over d_{max}. Once this length is reached the current path will not be explored any further.

3-7-3 Implementation

The stem is skeletonized using Zhang's algorithm as implemented in the scikit-image toolbox[3][39]. This iterative method removes border pixels until a single pixel wide representation remains. The obtained skeleton is analyzed with the Skan[4] library which produces graphs and the required branch statistics from skeleton images [40]. A skeleton is visualized in Figure 3-8a with junctions and tails. The irregular shape of the stem can cause the skeletonization

[3] https://scikit-image.org
[4] https://jni.github.io/skan/index.html

to create several small branches which are not actually there. This results in falsely identified junctions, and significantly increase the computational burden of the peduncle search. Therefore, small junction-tail type branches are removed using a minimum length threshold l_{min}. This threshold was determined experimentally and set to 10mm. As can be seen in Figure 3-8b many small branches have been removed without losing any relevant information, making the skeleton easier to analyze. Finally, the previously discussed peduncle search is applied with an angle threshold $\alpha_{\text{thresh}} = 45°$, and a maximum path length of $d_{\text{max}} = 12$. The resulting peduncle with junctions and tails is shown in Figure 3-8c.



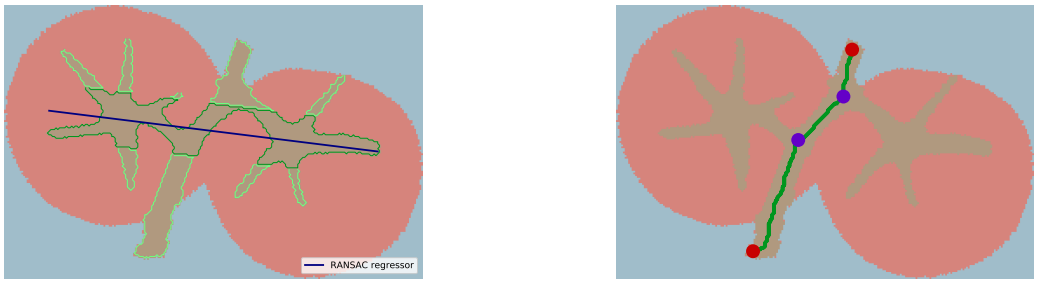
(a) Skeletonization of the stem green segment.

(b) Distance based thresholding of junction-tail type branches.

(c) Identified peduncle.

Figure 3-8: The peduncle detection process. Edges are marked by the dark green lines (—), junctions by the purple dots (●), and tails by the red dots (●).

In Figure 3-9 the newly developed graph-based peduncle detection method is compared with a Random sample consensus (RANSAC) regressor, which was used by Pekkeriet et al., 2015 [30]. By taking into account the peduncle curvature, the newly developed method works even for cases where the peduncle does not make up the largest pixel area present in the stem segment.



(a) RANSAC

(b) Graph-based method

Figure 3-9: The identified peduncle using two different methods. The graph-based peduncle detection is able to identify the peduncle in cases where the peduncle does not make up the larges pixel area in the stem segment.

3-8 Experiments and Results

The computer vision pipeline is applied to a dataset consisting of 84 labeled images, which contain 308 tomatoes and 266 junctions. The used parameters are shown in Table 3-1. The pipeline is deployed on an Intel i7-3630QM (2.40 GHz) processor. To evaluate the pipeline, we count the number of true positive predictions (TP), false positive predictions (FP), and false negative predictions (FN). The true positive rate (TPR) is computed, which describes the ability to find truss features present in the image:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3-14}$$

However, measuring the true positive rate is not sufficient, since the system may report many positives even if there are no features present. Hence, the false discovery rate (FDR) is used to determine the proneness of the system to classify negative examples as positive:

$$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}} \tag{3-15}$$

Furthermore, the errors of the estimated features are computed and reported as the mean absolute error plus-minus the standard deviation.

Table 3-1: Parameters used by the computer vision pipeline for the different stages.

stage	param	val	description
segentation	f	2	downsampling factor
	r	1.5	hue circle radius
	ϵ	0.01	k-means center movement stopping criterion
	$i_{\max}$	20	k-means iterations stopping criterion
filtering	d_{kernel}	3	circular kernel diameter in pixels
tomato detection	d_{blur}	3	Gaussian kernel size
	σ_{blur}	3	Gaussian kernel standard deviation
	$r_{\min}$	30	minimum tomato radius in millimeter
	$r_{\max}$	40	maximum tomato radius in millimeter
	$d_{\min}$	30	minimum tomato center distance in millimeter
	c_{dp}	4	inverse accumulator and image resolution ratio
	c_1	20	gradient threshold for accepting a pixel as an edge
	c_2	50	accumulator threshold for center and edge detection
	c_{fill}	0.5	minimum overlap between circle and tomato segment
peduncle detection	α_{tresh}	45°	maximum angle between previous path and the new edge
	$l_{\min}$	10	minimum branch length in millimeters
	$d_{\max}$	12	maximum number of edges in path

The tomato detection obtained a true positive rate of 100% (307/308). No false positives were reported. The tomato centers were predicted with an error of 3.58 ± 2.63mm (n=307) and the radii were estimated with an error of 2.43 ± 2.21mm (n=307). As a result, the center of mass is estimated with an error of 5.02 ± 3.26mm (n=307). Some results have been visualized in Figure 3-10.

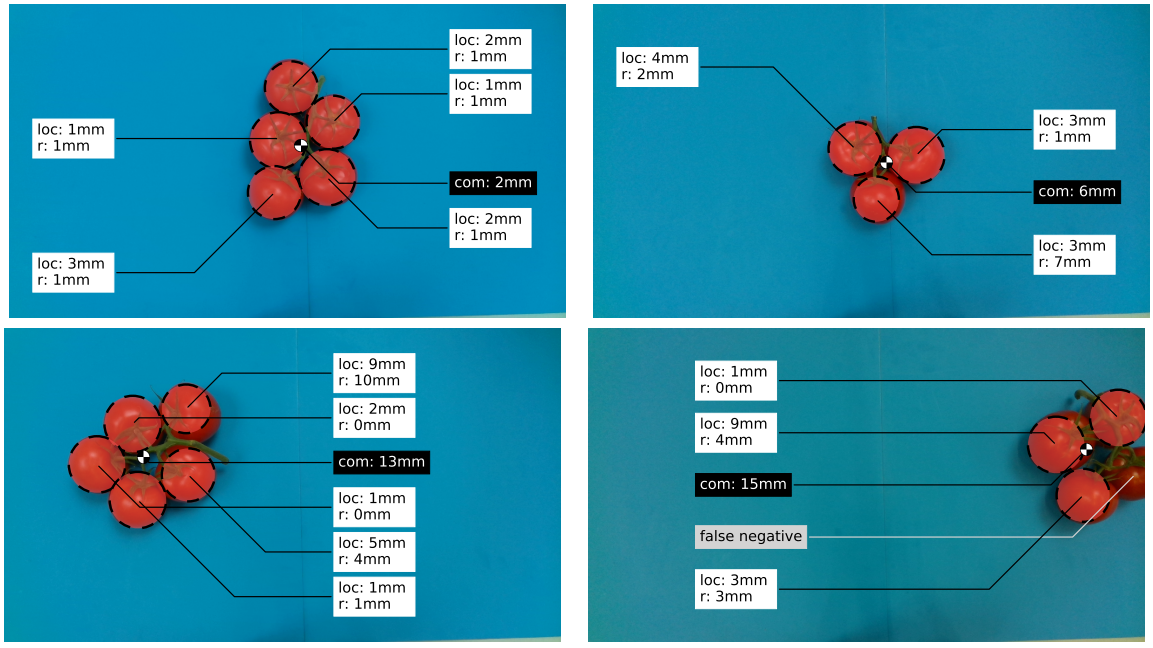


Figure 3-10: Images from the dataset, overlapped with the identified tomatoes. True positives are marked by the dashed circles (), and the determined center of mass by the crossed circle (). The values state the error made in determined location (loc) and radius (r). Generally, good results are obtained. Sometimes the radius of a tomato is underestimated, or a tomato is not detected affecting the predicted center of mass.

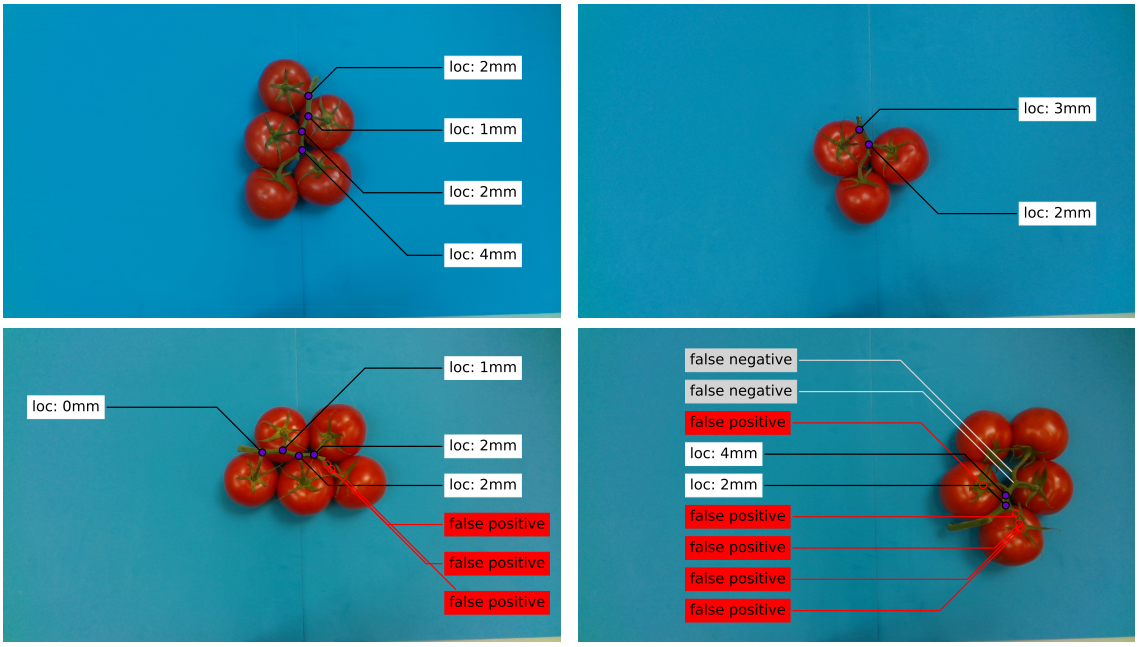


Figure 3-11: Images from the dataset, overlapped with the identified junctions. True positives are marked by the purple dots (), the corresponding value states the error made in determined location (loc). Most junctions are identified, but also several false positives are obtained. Sometimes the peduncle is not correctly detected, resulting in many false negatives and positives.

The junction detection obtained an overall true positive rate of 86% (229/266). Furthermore, a false discovery rate of 34% (117/346) was measured. The junction locations were predicted with an error of 2.09 ± 1.78mm (n = 229). Some results have been visualized in Figure 3-11. In Table 3-2 the error rates are reported for images where the truss is placed directly underneath the camera and where the truss is placed off-center. Both the true positive and false discovery rate are better on the images where the truss is placed at the center.

Table 3-2: Peduncle detection results split for images where the truss is placed in the center directly underneath the camera, and images where the truss is placed off-center.

truss placement	TPR	FDR	error [mm]
all	86% (229/266)	34% (117/346)	2.09 ± 1.78
center	96% (109/114)	29% (45/154)	1.68 ± 0.87
off-center	79% (120/152)	38% (72/192)	2.47 ± 2.24

Based on the obtained features, the target grasp pose is determined. The algorithm was able to find a target grasp pose on 96% (81/84) of the images. Some examples are given in Figure 3-12. The peduncle in the second image on the second row is misclassified. As a result, the target grasp pose is placed on the leaf of a calyx.

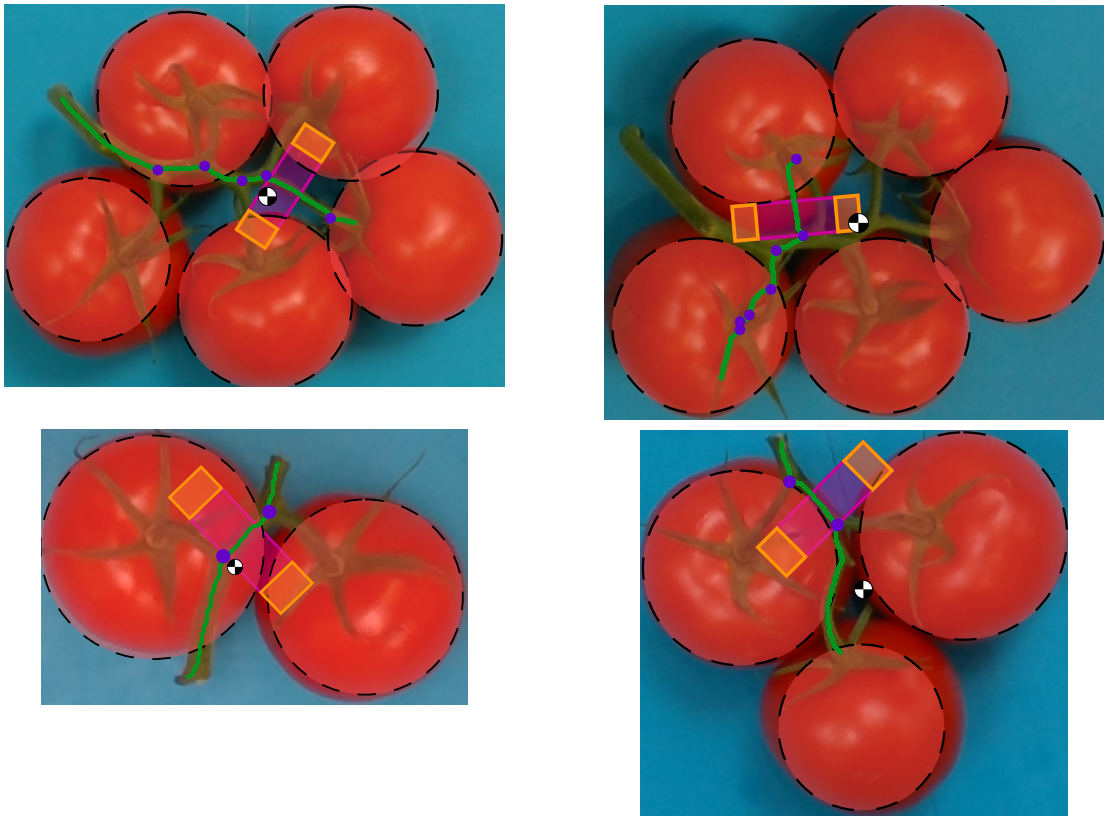


Figure 3-12: Several cropped images from the test set. The tomatoes are represented by the dashed circles (), the peduncle by the dark green line (), the junctions by the purple dots () the center of mass by the crossed circle (), and the opened end effector fingers by the orange boxes (), which will be close along the purple lines ().

The computer vision pipeline took 1.88 ± 0.83 seconds to execute per image. Figure 3-13 shows the average time taken by each part of the pipeline. Peduncle detection, cropping and segmentation take up a significant amount of time. All images were processed between 0.74 and 4.17 seconds.

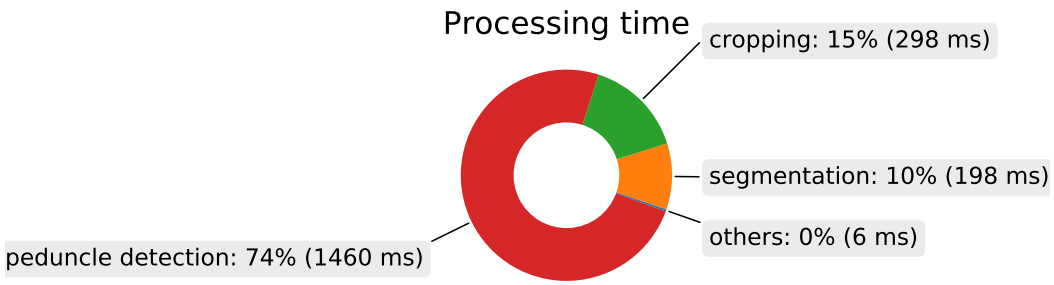


Figure 3-13: Average processing time of the different computer vision pipeline stages.

3-9 Discussion

The pipeline is able to identify almost all tomatoes without reporting any false positives. The center of mass estimate is sensitive to the prediction errors of the tomato features. Two factors affect this prediction error. First of all, the tomato features are harder to estimate when a tomato is closely surrounded by others or at the edge of the image. For these tomatoes the circle can only be fitted to a small part of its edge. Secondly, the fitted parameters are affected by the ellipsoidal shape of the tomato, sometimes the circle is fitted to a part of the edge which has a smaller or larger radius than the tomato. The circle fitting could be improved by utilizing the edge of the individual tomatoes, instead of using the edge of the entire segment. Benavides et al., 2020 [29] combine color-based segmentation with fruit edge detection to identify individual tomatoes on a cluster. Fitting could be improved further by using more complex shapes such as ellipses, as done by Pekkeriet et al., 2016 [11].

The proposed graph-based peduncle search method is able to correctly identify the peduncle in cases where a previously proposed RANSAC regressor fails. The newly developed method works best when the camera is placed directly above the truss, the performance degrades when the object is placed off-center. Many junctions are detected, but also many false positives are reported. The junction detection is sensitive to overlapping parts of the stem and small segmentation errors. Furthermore, the graph-based peduncle search often classifies a pedicel and even parts of a calyx as peduncle, resulting in many false positives. Performance could possibly be improved by adding more assumptions on the peduncle shape, such as maximum number of junctions and minimum distance between these junctions. Furthermore, if the calyxes would be identified and removed before performing the graph-based peduncle search, the amount of false positives could be reduced significantly.

3-10 Summary and Concluding Remarks

A computer vision pipeline for vine tomato was presented and evaluated. The images are segmented by applying k-means clustering on the hue and a* color components. Tomatoes are detected using the CHT on the tomato segment. A graph-based peduncle search method is introduced to identify the peduncle and junctions on the skeletonized stem segment. The entire pipeline uses 17 parameters. Many are intuitive to tune, or do not impact the results as long as they are in a reasonable range. However, some are difficult to tune since they depend on others. For example, the accumulator threshold is affected by the accumulator resolution and tomato radius and can therefore not be tuned individually.

Improvements are required to employ this pipeline in real-world scenarios. Three recommendations will be given. First of all, the processing time of approximately two seconds per truss is too long. Improvements can be made since time efficiency was not the aim of this work. The peduncle search may be improved by filtering irrelevant edges before performing the search. A possible criteria would be to identify the calyx and remove all attached edges. Also the search from different origin vertices can be parallelized, and memory can be added such that identical paths are only traversed once. Furthermore, the cropping time may be reduced by placing a rotated bounding box directly, and thus no longer rotating the entire image. Finally, the segmentation time can be reduced by reusing previously determined cluster centers instead of recomputing them on each run.

Secondly, the color based segmentation only works when the background, tomato and stem have a unique color. The segmentation will misclassify green tomatoes and fail when there are other plants on the background. Furthermore, the used segmentation cannot distinguish different touching truss instances in an image. Mask R-CNN, as presented by He et al., 2017 [41], is an advanced segmentation method which is able to identify for each pixel the object instance it belongs to. It has been applied to strawberry [42] and leaf detection [43, 44]. Alternatively, instances can be differentiated after segmentation. Luo at al., 2018 [45] propose a method for separating double overlapping grape clusters based on their geometry. However, no method for determining whether a given binary image contains overlapping grapes is given.

Thirdly, occlusion naturally arises in many situations, for example when trusses arrive in a multi-layered harvest crate. Tomato detection will fail in this case, since the tomatoes are surrounded by others, leaving no edge to fit circles on. Identifying the different instances becomes more difficult in this case, since many trusses will be partly visible. For such cases, depth information may be utilized to separate the top layer in the crate from the lower layers. Lin et al., 2020 [46] propose a depth image clustering algorithm which generates a set of clusters from depth images. A three-dimensional shape detection algorithm is used to fit multiple spheres or cylinders on the clusters.

Chapter 4

Experiments and Results

Real-world experiments were conducted to validate the proposed geometry-based grasping method for vine tomato. These experiments were performed in a well controlled lab environment. In this chapter the experimental setup, its calibration, the pick and place routine, and the results are discussed.

4-1 Experimental Setup

The physical setup consists of a robotic manipulator, an RGB-D vision system and several tomato trusses. The manipulator is used to perform truss grasping routines, while the vision system provides the required information to create a model of the truss. An overview of the entire experimental setup is shown in Figure 4-1.

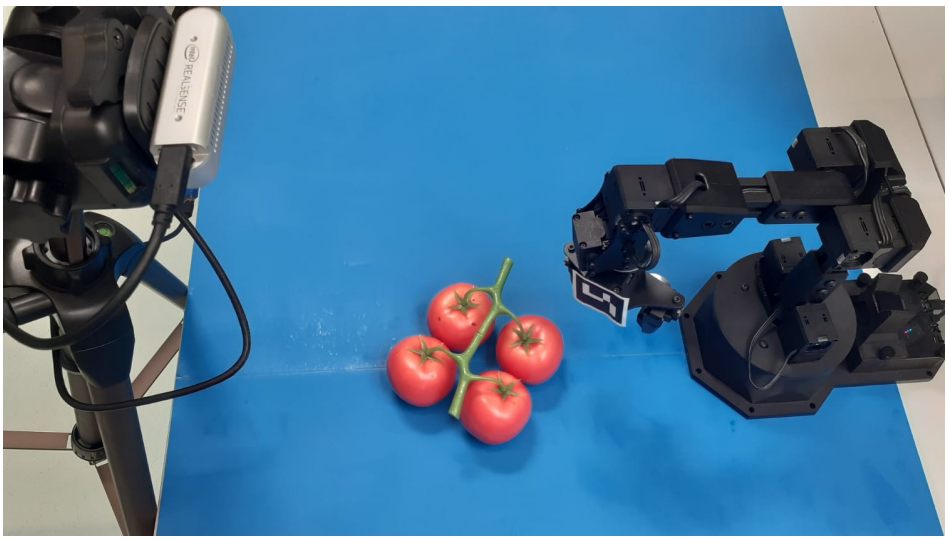


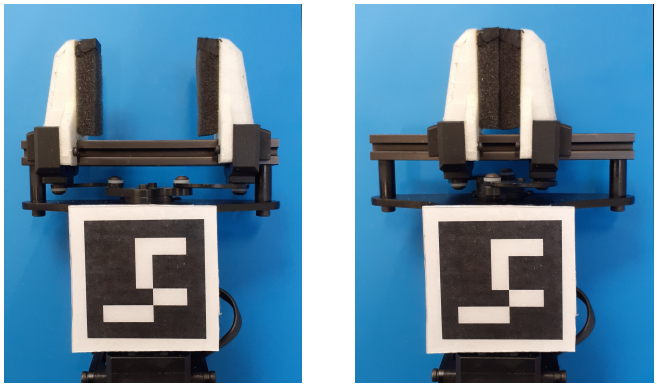
Figure 4-1: An overview of the experimental setup.

4-1-1 Manipulator

The Interbotix PincherX 150 Robot Arm is used for manipulation. Because of its small size it is not bounded to a lab environment, which was required due to covid-19 restrictions. The five degrees of freedom are actuated by eight DYNAMIXEL XL430-W250-T Smart Servo Motors, which offer a maximum working payload of 50g [47]. The robot can reach in full 360° rotation up to 450mm from its base, at a 5mm accuracy. End effector fingers of desired dimensions were 3D printed. The inside of these fingers are covered with deformable foam as required for the soft-part caging condition. Furthermore, rubber knobs were attached to these fingers such that they can enclose the peduncle. The end effector is shown in a open and closed configuration in Figure 4-2, the dimensions are given in Table 4-1. A more extensive motivation for using this manipulator and a comparison with other manipulators can be found in Appendix C.

Table 4-1: Finger dimensions of the custom designed end effector.

parameter	value [mm]	description
h	43	height
w	10	width
h_{tip}	7	height of the finger tip
t_{tip}	8	thickness of the finger tip



(a) open configuration **(b)** closed configuration

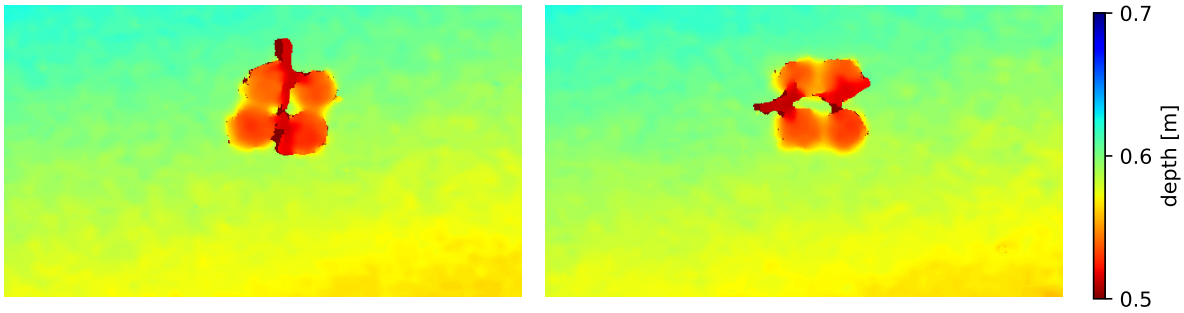
Figure 4-2: The modified end effector. The custom fingers are 3D-printed, the finger tips are created from rubber knobs, and the inside of the fingers are covered with deformable foam.

4-1-2 Vision

The Intel RealSense Depth Camera D435[1] combines active infra-red (IR) stereo technology to measure depth at a resolution up to 1280 × 720 pixels with an RGB sensor for color information at a resolution up to 1920 × 1080 pixels. Since a sense-plan-act framework is used, the camera is placed in an eye-on-base configuration. To improve the depth output, post-processing is applied on the host computer. A temporal and spatial filter are applied in

[1] https://www.intelrealsense.com/depth-camera-d435/

the disparity domain, the result is shown in Figure 4-3a. Furthermore, upon computing the three-dimensional location of a point, non-zero median sub-sampling with a factor of five is used as recommended by the manufacturer [48]. Despite these efforts, the camera in unable to obtain depth information about the peduncle when the truss has a horizontal orientation. This is shown in Figure 4-3b. Therefore, the peduncle height is taken as a constant, measured before conducting an experiment.



(a) Truss with horizontal orientation. **(b)** Truss with horizontal orientation.

Figure 4-3: Visualization of the recorded depth, for a truss placed in a vertical and horizontal orientation.

4-1-3 Target Object

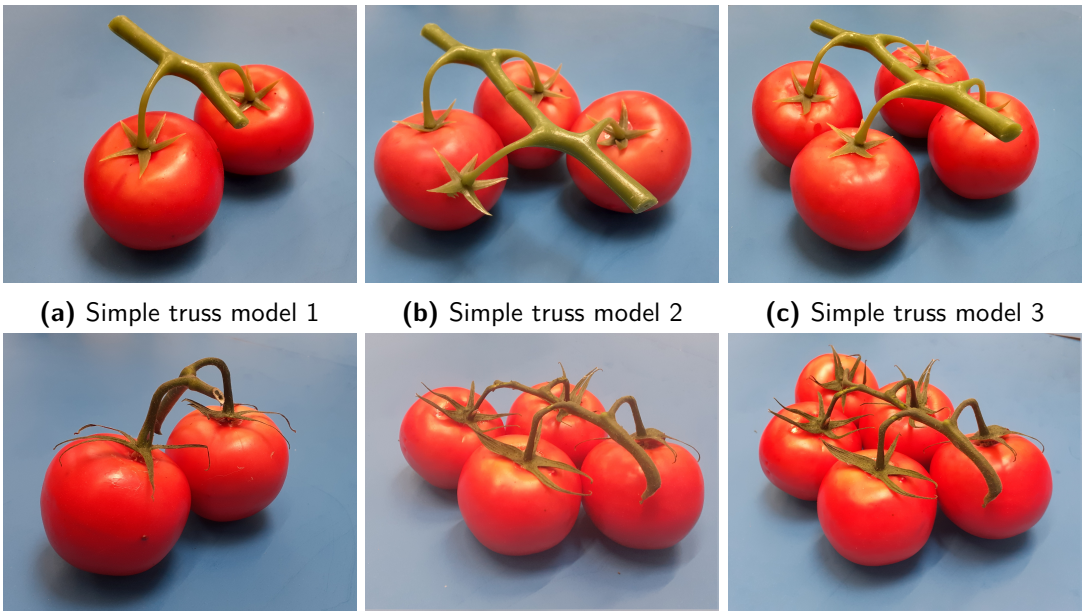
Payload limitations of the manipulator do not allow for executing pick and place tasks on real tomato trusses. Plastic models offer an interesting alternative since they are much lighter and may be modified to create trusses of various levels of realism and grasping complexity. The original plastic model is shown in Figure 4-4a. From these models two different truss categories are created:

1. **Simple truss**: artificial tomatoes are attached to an artificial peduncle. This peduncle lies 2 cm above the tomatoes, has a large diameter and a simple straight shape. Three models were created as shown in Figure 4-4a to 4-4c.
2. **Realistic truss**: artificial tomatoes are attached to a real stem. This gives a realistic truss geometry with slightly less variation between the tomatoes. Trusses of various shapes were created, as shown in Figure 4-4d to 4-4f.

Truss stems with sufficient space around the peduncle to fit the gripper fingers were selected.

4-1-4 Communication and Control

All parts are integrated via Robot Operating System (ROS). The manipulator is connected to a computer via USB with the U2D2. This device converts USB signals to TTL which allows communication with each of the Dynamixel servo motors. The driver for the manipulator is build on top of the Dynamixel Workbench Toolbox, a C++ library containing functions to command and get feedback from the motors. This driver and several other packages containing



(a) Simple truss model 1 **(b)** Simple truss model 2 **(c)** Simple truss model 3

(d) Realistic truss model 1 **(e)** Realistic truss model 2 **(f)** Realistic truss model 3

Figure 4-4: Overview of the truss models used for the pick and place routines.

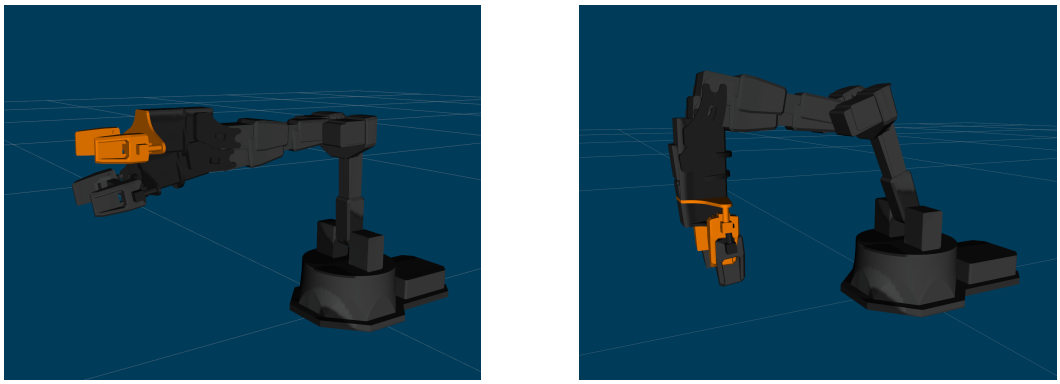
the robot description, simulation, and MoveIt support can be found in their repository[2]. The camera communicates with the computer over USB. The provided ROS wrapper by Intel RealSense is used[3].

The manipulator uses a PID controller with velocity and acceleration feed-forward terms to control the angle of each joint. The trajectory setpoints are generated via MoveIt's trajectory planning and inverse kinematics. The torque provided by the motors is limited for the manipulator weight. As a result, large static control errors are observed for poses which require high joint torques as shown in Figure 4-5. To decrease this error, the target pose height is adjusted upwards with an amount proportional to the distance between the target pose and the robot base. The gripper is PWM controlled to limit the maximum applied force. This prevents overload errors on the gripper motor when it is blocked by an object. Position feedback from the gripper is used to determine whether a desired gripper configuration is obtained.

4-2 Calibration

Calibration is required to obtain the transformation between the object and camera, the camera and robot base, and between the robot base and hand. These transformations are shown in Figure 4-6. Intrinsic, extrinsic, and manipulator calibration are necessary to obtain these three transformations. The manipulator calibration contains the mapping of encoder ticks to joint angles of the robotic manipulator, this is already present in the robot's firmware.

[2] https://github.com/Interbotix/interbotix_ros_arms
[3] https://github.com/IntelRealSense/realsense-ros



(a) Home configuration **(b)** Grasp configuration

Figure 4-5: Control error for two different poses, the orange gripper shows the target pose, while the grey gripper shows the actual pose.

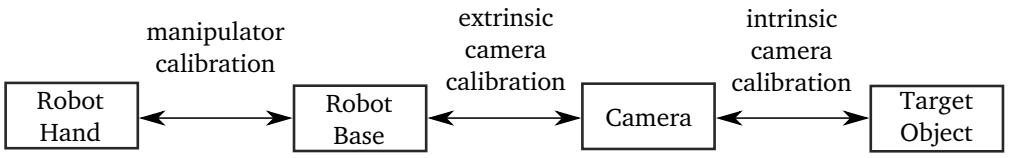


Figure 4-6: A diagram containing the different objects which need to be calibrated relatively to each other, with the corresponding calibration type.

4-2-1 Intrinsic Calibration

The camera calibration includes parameters, such as the focal length and lens distortion, but also the transformations between the RGB, right IR and left IR sensor. The D435 camera comes calibrated, but this calibration may degrade over time due to exposure to temperature cycles, shock and vibration. Recalibration can be performed using on-chip calibration[4]. A well textured flat target is used with the projector turned off as recommended by the manufacturer.

4-2-2 Extrinsic Calibration

Extrinsic calibration is used to determine the transformation which describes the pose of the camera relative to the robot base. The method proposed by Tsai and Lenz, 1989 [49] is used. The end-effector is moved to different poses where samples of the robot base and camera pose relative to the end-effector are taken. The pose of the camera relative to the end-effector is determined by placing an Aruco marker on the end effector. The placement of the marker is taken as an additional unknown, and thus does not need to be measured manually.

The original work provides an error analysis to determine crucial factors during calibration which influence the accuracy. One should aim to maximize angles between different rotation axes. Furthermore, the rotations between each pose should be maximized, but the translations should be minimized. Additionally, it is desired to minimize the distance between the camera lens and the tracking marker, and to use redundant poses. Finally, the quality of the result is

[4] https://www.intelrealsense.com/self-calibration-for-depth-cameras/

affected by the camera intrinsic and robot calibration. Therefore, the implemented calibration procedure uses two yaw and roll angles of $-40°$ and $40°$, creating four poses. These are recorded at two different pitch angles, resulting in a total of eight poses.

4-3 Pick and Place Routine

The aim of the experiments is to determine whether the proposed method can successfully be used to grasp vine tomato. Therefore, a pick and place routine is executed repeatedly to determine the strengths and weaknesses of this method. The task consists of five successive parts: (i) the truss is detected and relevant features are extracted, (ii) the end effector moves towards and wraps around the peduncle, (iii) the truss is lifted off the supporting surface, (iv) the truss is moved to an arbitrary target pose, and (v) the truss is placed back on the supporting surface. This task is successful if all these parts are completed. The following modes of failure may be observed:

- **Vision**: the vision system fails to identify key features required for the geometric model, such as the peduncle, or junctions. These failures are caused by the presented computer vision pipeline.
- **Planning**: the grasp planning algorithm fails to determine a valid grasp action. For example, a pose which can not be reached by the end-effector due to space constraints. This error shows weaknesses in the proposed geometry-based grasping method.
- **Control**: the manipulator fails to execute the planned task, resulting in the truss not being picked or dropped while executing a task. A typical example is the end effector not reaching the target pose. These errors show weaknesses in the used hardware, calibration, and implementation such as used filters.
- **Overload**: the hardware fails to execute the task at hand due to overloading or overheating of the servo motors.

In case the task is not successfully executed, the mode of failure will be noted for later discussion. Since the goal of this experiment is not to test the load capacities of the robot, overload failures will be ignored. When a grasp fails the truss is moved to a new random pose by the human operator such that the experiment can continue. The pick and place task will be executed for the two previously described truss types.

4-4 Results

The previously described pick and place routine is performed 20 times per truss, such that the truss needs to be grasped from several different poses. The computer vision pipeline settings are stated in Table 3-1. The failure rates are provided in Table 4-2, some grasps are shown in Figure 4-7. Videos taken during the experiment can be found via SurfDrive[5].

[5] https://surfdrive.surf.nl/files/index.php/s/euEKKhtSRqB9mPG

Table 4-2: Failure rates of the proposed geometry-based grasping method applied to various truss models. Two different model categories were used. The simple models use an artificial stem, whereas the realistic models consist of a real vine tomato stem.

			failure rate			
category	model id	attempts	all	vision	planning	control
simple	**all**	**60**	**8% (5)**	**3% (2)**	**3% (2)**	**2% (1)**
	1	20	5% (1)	5% (1)	0% (0)	0% (0)
	2	20	20% (4)	5% (1)	10% (2)	5% (1)
	3	20	0% (0)	0% (0)	0% (0)	0% (0)
realistic	**all**	**60**	**17% (10)**	**0% (0)**	**7% (4)**	**10% (6)**
	1	20	0% (0)	0% (0)	0% (0)	0% (0)
	2	20	25% (5)	0% (0)	20% (4)	5% (1)
	3	20	25% (5)	0% (0)	0% (0)	25% (5)

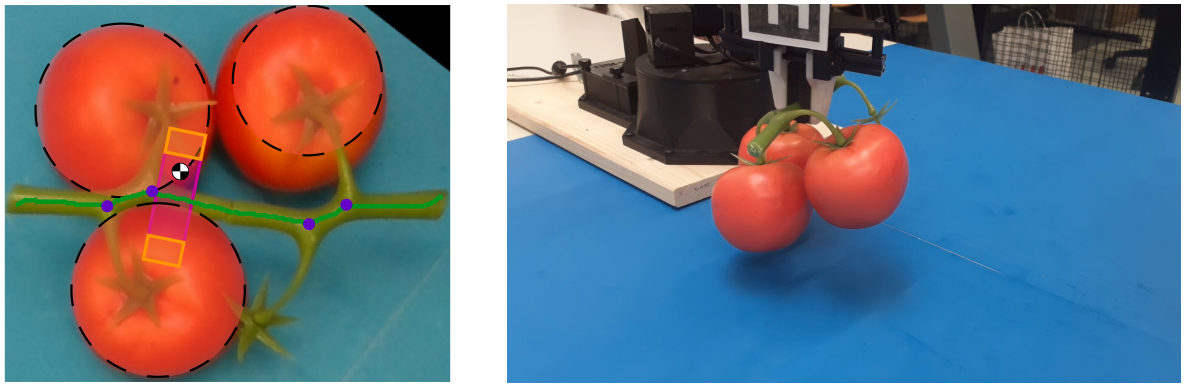
A failure rate of 8% (5/60) was obtained on the simple models. A typical successful grasp is shown in Figure 4-7a. From the five failures, two were caused by vision, two by planning, and a single failure was caused by control. For both vision failures, the algorithm was unable to identify the peduncle. In both planning errors, the algorithm did not take into account the surrounding pedicels, causing the end effector to be blocked. An example is shown in Figure 4-7b. The other failure was caused by control, where the end effector closed above the peduncle. Most failures occurred on model two, a single failure was made on model one, and no failures were made on model three. Undesired tilting of the second model was frequently observed as shown in Figure 4-7c.

On the realistic models, a failure rate of 17% (10/60) was obtained. Twice as many failures were reported compared to the simple models. All ten failures were caused by pedicels blocking the end effector. Four of these cases were caused by planning, and six cases were the result of control. No failures were caused by the computer vision system. All grasp attempts on the first model succeeded, an equal number of failures was observed for model two and three. Most failures on model two were caused by planning, on model three control errors occurred more frequently.

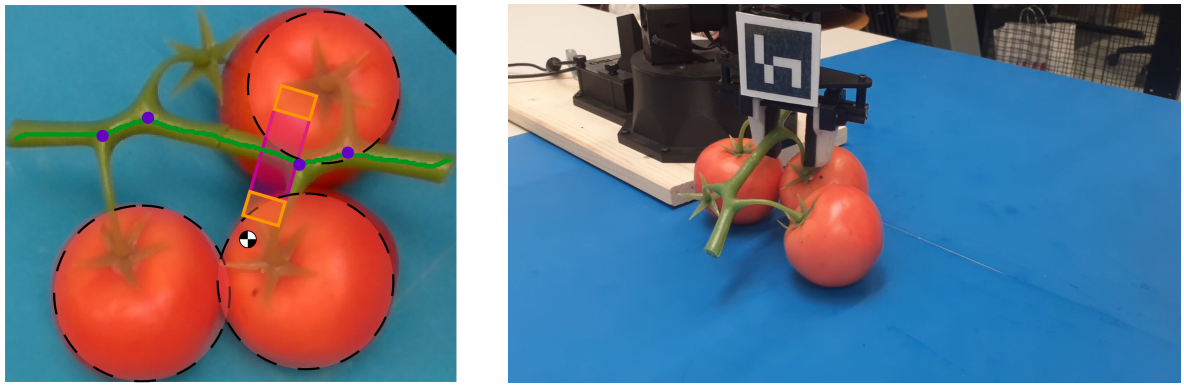
4-5 Discussion

In the presented experiments, a significant number of errors were caused by the manipulator control. However, not all failures may be attributed to the hardware. For trusses that require more accurate grasping a significant amount of planning errors are made. On the third simple model the center of mass, and thus also the target grasp pose, is close to a pedicel. At such a location small errors in planning can result in grasp failure, which explains the lower success rate for this model. The realistic models have less space around the stem, and the pedicels are located closer to each other. This makes the planning and control of a successful grasp more difficult for these models than the simple models.

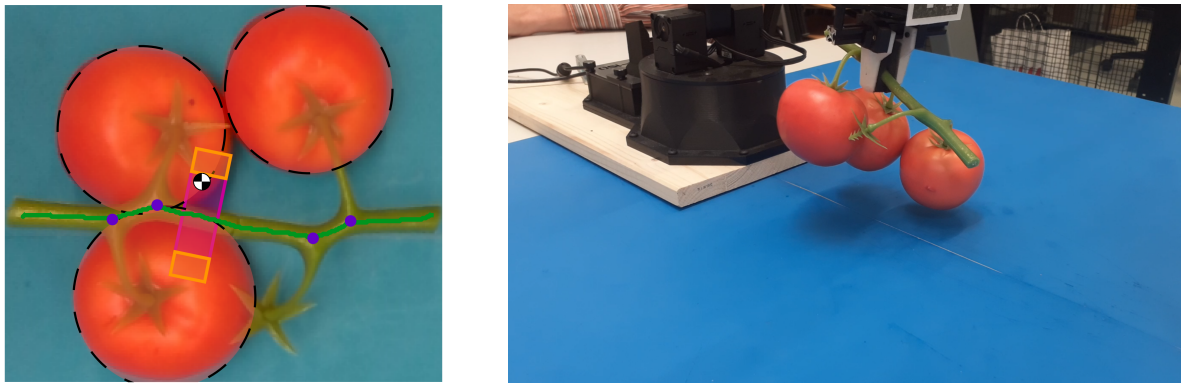
The grasp pose is determined based on several criteria, among which the space available at the target location on the peduncle. However, it ignores surrounding stem parts and tomatoes which may block the grasp. The used two-dimensional truss model can not be used to avoid



(a) A successful grasp where the end effector fingers wrap around the peduncle.



(b) A failed grasp where the planned closing actions causes the end effector fingers to get stuck on the pedicel.



(c) The model tilts during a grasp.

Figure 4-7: A pick and place routine executed on simple truss model two. The left column shows the planned grasp, and the right column shows the grasp applied to the truss. The tomatoes are represented by the dashed circles (), the peduncle by the dark green line (), the junctions by the purple dots () the center of mass by the crossed circle (), and the opened end effector fingers by the orange boxes (), which will be close along the purple lines ().

these surrounding objects, as depth information is vital. To make a more appropriate trade-off, the point cloud of the tomato truss should be analyzed instead of computing a target grasp location based on a two-dimensional image. Furthermore, a grasp is always approached in a vertical direction as it is assumed that the truss lies on a flat surface. By allowing for different end effector orientations and approach directions, the end effector may grasp the peduncle at previously unreachable locations. The collision detection and contact determination system from GraspIt! can be used to identify a target grasp location on a given point-cloud [50]. For implementation, the caging-based grasping constraints need to be included and a grasp quality metric based on the amount of free space and distance to the truss center of mass. One of the challenges is to obtain an accurate point cloud of the tomato trusses.

The used experimental method has its limitations. First of all only a few trusses were used, no conclusions can be drawn about the success rate of the proposed method on vine tomato in general. Secondly, only truss stems with sufficient space around the peduncle to fit the gripper fingers were used. Trusses exist where the peduncle lies much closer to the tomatoes. For such trusses, the proposed geometry-based method will be much more difficult if not impossible to apply. Finally, due to hardware limitations, artificial tomatoes were used for the experiments. However, using these artificial tomatoes may affect the experimental results. First of all, the unrealistic artificial tomato weight may affect the obtained results. The truss stem does not bend as it would under the weight of real tomatoes, and the assumption that the peduncle mass is negligible compared to the tomato mass does not hold for these lightweight tomatoes. As a result, the predicted center of mass is off, causing the truss to tilt. Secondly, the artificial tomatoes encompass much less geometric variety than the real fruit, possibly making the grasping task less complex. Finally, the damage done to these fragile objects could not be measured, which is an essential aspect to automate the packing task of vine tomato.

Chapter 5

Conclusion

In this thesis, a geometry-based grasping method was developed to grasp vine tomato. This method solely relies on geometric information and position-based control. A geometric model of both vine tomato and the end effector was developed. These models were used to derive concrete constraints for a caging-based grasp and determine an optimal grasp pose. A computer vision pipeline was developed to determine the numerical values of several geometric truss features. Finally, the proposed computer vision pipeline and the geometry-based grasping were tested on a robotic setup.

The computer vision pipeline is able to obtain key features for trusses of various shapes and sizes. On the constructed dataset, 100% of the tomatoes were detected, without reporting any false negatives. Prediction errors on the tomato locations and dimensions result in an error of 5.02 ± 3.26mm on the truss center of mass. The graph-based peduncle detection searches for the longest path on the stem with limited curvature. It was shown, that this new method can identify the peduncle correctly on trusses for which previous methods have failed. The algorithm correctly predicted 86% of the locations where the stem splits, but also 34% false positives were reported. The real-robot experiments show that the newly developed grasping method is capable of grasping vine tomato. Experiments were performed on simple trusses consisting of an artificial stem, and realistic trusses constructed from a real vine tomato stem. On the simple models, 92% of the pick and place attempts succeeded and 83% on the realistic models.

This is a first step in the direction of automating the packing task of vine tomato. However, future studies are needed to develop extensions required for real-world implementation. Experiments should be performed on more high-end hardware and real tomatoes, such that the damage to the crop can be analyzed. The presented method does not take into account the surrounding stem parts or tomatoes when computing a target grasp location. A point cloud of the tomato truss should be used such that these surrounding objects can be avoided. Other extensions are to conduct experiments on a larger batch of trusses and deal with multiple touching and overlapping instances.

Appendix A

Dataset

A dataset is created to evaluate the developed computer vision pipeline. This appendix discusses the recorded color and depth data, the labeling process, and how these labels are compared with the predictions to identify true positive, false positive and false negative estimates.

A-1 Color Data

From each truss 7 images are taken. For the first three images the truss is placed near the center at an horizontal, vertical and random orientation. for the fourth and fifth image, the truss is placed slightly off the center at a random orientation. Finally for image 6 and 7 the truss is placed near the edge of the image border at a random orientation. The trusses used were bought at the Lidl.

A-2 Depth Data

To convert the parameters expressed in millimeter to a dimension expressed in pixels, a constant is determined which approximates the amount of pixels per mm. Since the camera is placed perpendicular to the table the distance between the camera and table is estimated using the recorded depth information $\boldsymbol{z}$ as follows:

$$d = \text{median}(\boldsymbol{z}) \tag{A-1}$$

Than the amount of pixels per millimeters can be estimated using the simple camera model

$$x = \frac{f}{d}X \tag{A-2}$$

Table A-1: An overview of the constructed data set. Six different clusters have been used, which have been modified by splitting them and removing tomatoes. Per truss seven images are taken, where the truss is placed at several different poses.

image ID	cluster ID	tomatoes	junctions	conditions
1 - 7	cluster 1	5	4	full truss
8 - 14	cluster 2	5	4	full truss
15 - 21	cluster 3	5	4	full truss
22 - 28	cluster 1a	2	2	split truss
29 - 35	cluster 1b	3	2	split truss
36 - 42	cluster 2a	4	4	missing tomato
43 - 49	cluster 3a	4	4	missing tomato
50 - 56	cluster 4	5	5	full truss
57 - 63	cluster 5	5	5	full truss
64 - 70	cluster 6	2	2	split truss
71 - 77	cluster 4a	2	2	split truss
78 - 84	cluster 5a	2	2	split truss

with the true dimension X in mm, dimension as it appears in the image x in pixels and focal length f in pixels. Since the focal length is not exactly identical in x and y they are averaged,

$$f = \frac{f_x + f_y}{2} \tag{A-3}$$

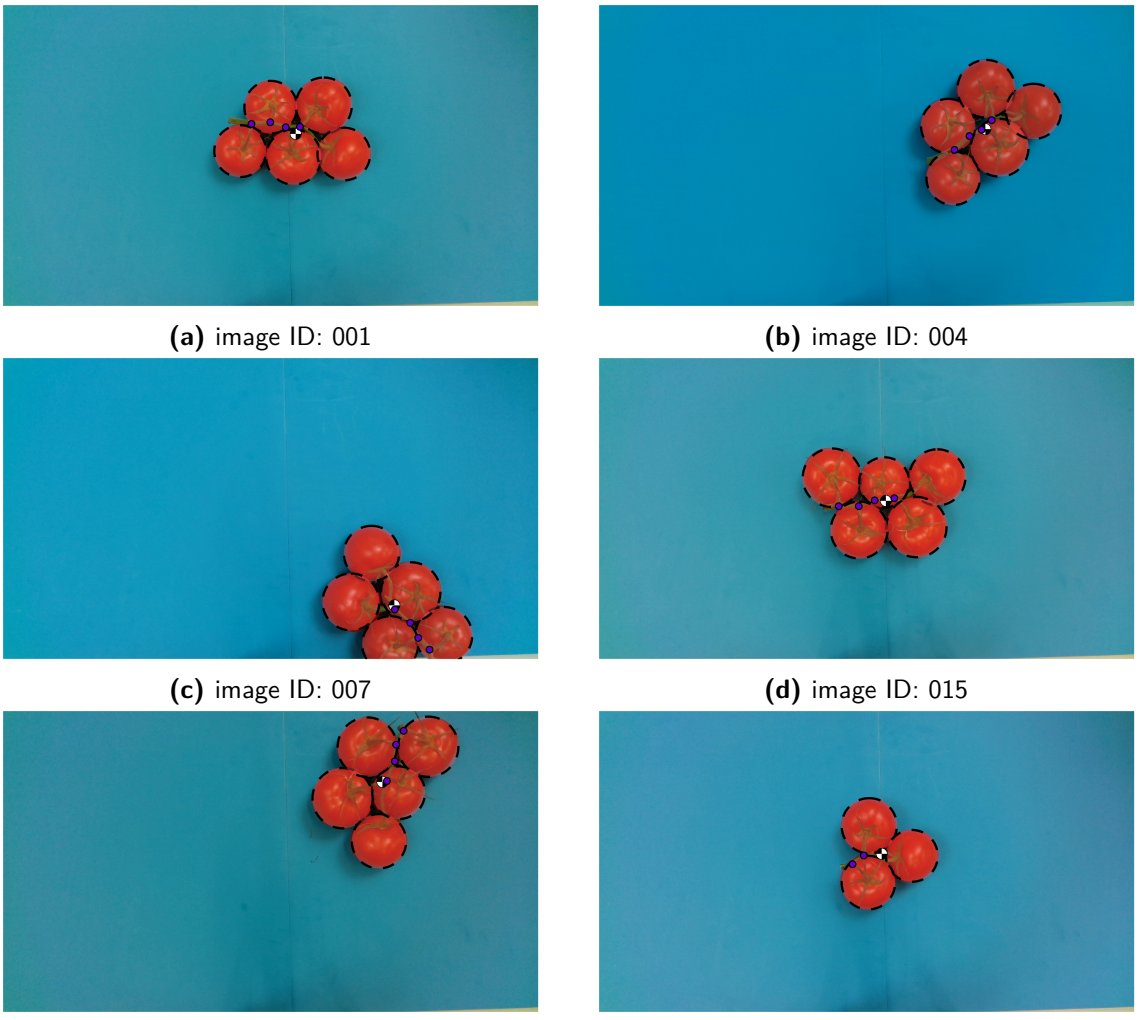
The tomatoes are close to the camera than the table, due to the height of a truss. Therefore, an error will be made when converting distances. It is expected that lies lies within a 15% range.

A-3 Labeling

The images are labeled using the graphical image annotation tool Labelme [51]. The tomaotes are label with circles, and the junctions with points. Some labeled images are shown in Figure A-1. The center of mass is calculated based on the provided tomatoe labels

A-4 Verification

To determine the performance of the algorithm the predictions are labeled as true positive, false positive and false negative. In the case where we only have true positives, each prediction is assigned to the closest label. Furthermore, the predictions have to lie within a minimum distance distance of the label, otherwise the prediction is marked as a false positive. This minimum distance is 15mm for tomatoes and 10mm for junctions, since predictions large errors than these will be useless. In case we can not assign all predictions to labels, the remainders will be stated to be false positives. If certain labels are not assigned to any predictions than these are stated to be false negatives.



(a) image ID: 001 **(b)** image ID: 004

(c) image ID: 007 **(d)** image ID: 015

(e) image ID: 020 **(f)** image ID: 030

Figure A-1: Labeled data, the tomatoes are marked by the dashed circles (), the junctions by the purple dots (), and the center of mass by the crossed circle ().

Appendix B

Color Spaces

The human eye has three cone types which captures long, medium and short wavelengths. These somewhat coincide with red, green and blue colors. Therefore three values, which compose a tristimulus, can be used to describe any visible color. In the original image taken by the camera, the color is described using red, green and blue (RGB) values. Many alternative descriptions may be used. The performance of the image segmentation depends on the choice of color space. An overview of some commonly used color spaces is provided as discussed by Busin et al., 2008 [34]. These color spaces are used to describe the image shown in Figure B-1, the separate color components are visualized in Figure B-2.

Figure B-1: Original image taken by the camera.

B-1 RGB

The combination of colors represents how red, green and blue light are added together considering a black base color. It models the output of many physical devices. Therefore, the main purpose of this color space is sensing and representation of images in electronic systems. Humans do not perceive color as a combination of three primary colors, but according to more subjective entities related to luminosity, hue, and saturation. Alternative color spaces have been designed, which align more closely with the human perception of colors. They may be obtained by either applying a linear or non-linear transformation to the RGB components.

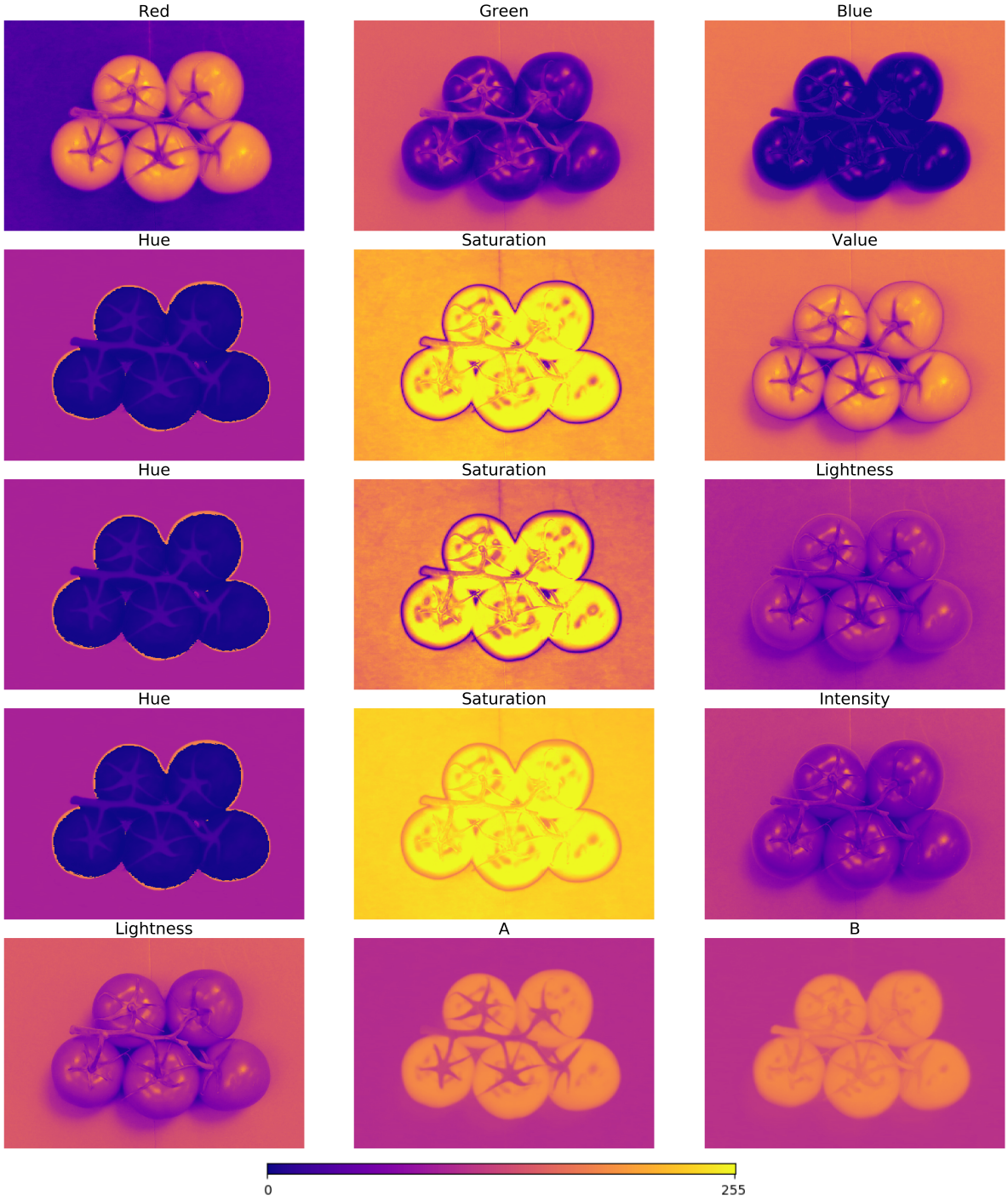


Figure B-2: The different color components of an image, represented in various color spaces. The three columns contain the color components of each color space. The Hue value is scaled down by a factor two to fit the [0,255] range. The values of LAB have been rescaled to lie in the 0-255 range

B-2 HSI, HSL and HSV

Contrary to the cartesian RGB color space, these are polar coordinate spaces. The angular dimension of represents the hue, which starts at red (0°), and passes through green (120°), and blue (240°) and wraps back to red (360°). The vertical axis represents luminosity for which several definitions exist depending on the purpose and goals of the representation: (i) intensity (I) is the average of the three color components, (ii) value (V) is the largest component of a color and (iii) lightness (L) is the average of the largest and smallest color components. The radius of the cylindrical coordinate is the saturation axis which is related to the chroma. This chroma represents color intensity, and the obtainable values are dependent on the lightness. This is undesirable for certain applications, such as color selection tools. Therefore hue, saturation and value (HSV) and hue, saturation and lightness (HSL) define the saturation as a scaled chroma such that its range is independent of lightness or value. hue, saturation and intensity (HSI) however is designed to facilitate separation of shapes in an image. Therefore saturation is defined as chrome relative to lightness which is more in line with human perception.

B-3 XYZ

Most wavelengths stimulate two or all three kinds of cone cell because the spectral sensitivity curves of the three kinds overlap. Certain tristimulus values are thus physically impossible. Furthermore, LMS tristimulus values for pure spectral colors would, in the RGB color spaces, imply negative values for at least one of the three primaries. The XYZ system is introduced to avoid these negative RGB values, and to have one component that describes the perceived brightness. It uses three imaginary primary colors and may be transformed to the RGB space using a linear transformation. These imaginary colors do not correspond to any spectral distribution of wavelengths and therefore have no physical reality. This space is perceived as non-linear by humans. That is, in some parts of the space, huge color changes are produced by little position variations; while in other parts of the space happens the opposite, little color changes are experimented for large position changes. Thus, variations of the CIE model have been developed for different kinds of applications, or to represent the colors such that it mimics the human perception of color.

B-4 L*a*b*

L*a*b* is based on a non-linear transform of the XYZ color space. It was designed such that the same amount of numerical change in these values corresponds to roughly the same amount of visually perceived change. The L* component lies in the range 0 to 100 range and describes the lightness from black to white, the a*-component describes the distribution of colors from red to green, and the b*-component describes the distribution of colors from blue to yellow. A neutral color is represented by a value of zero, and thus the values may take up negative and positive values.

Appendix C

Manipulator

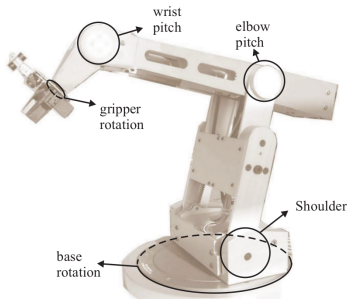
For the real-world experiments, a manipulator should be used which meets a set of requirements. In this appendix, a list of requirements is created, and a suitable manipulator is chosen.

C-1 Requirements

The manipulator should be able to move a two fingered gripper towards, and parallel to, the truss peduncle. The manipulator should be deployable at home, since COVID-19 restrictions might hinder lab access. An environment has already been set up in ROS to command the hardware via MoveIt! Creating an ROS packages for MoveIt! can be time consuming and complex, therefore MoveIt! support is desirable. A complete truss of tomatoes can easily weigh more than 500g, which is an unrealistic requirement for these small manipulators. Alternatively, we may lift artificial vine tomatoes which weigh about 50g per pair. In this case we require the manipulator to lift at least 50g. Finally, due to the Corona measures the delivery of parcels is often delayed, or sometimes delivery is not possible at all[1]. Therefore, it is desired that the manipulator is located in a warehouse near the Netherlands. To summarize, the manipulator must:

- Have at least 4 DoF
- Have a wrist joint to control the yaw of the end effector.
- Come with a two fingered gripper
- MoveIt! support
- Handle at least a 50g payload
- Be shipped from a warehouse near the Netherlands.
- Should cost less than €1,000.00[2]

[1] https://www.postnl.nl/en/customer-service/coronavirus/frequently-asked-questions/
[2] All prices given are including VAT.



(a) Ed-Ro manipulator, taken from [52].

(b) RAMRobot, taken from [53].

(c) uArm from UFACTORY

(d) OpenMANIPULATOR-X by ROBOTIS

(e) PhantomX Reactor by Trossen Robotics.

(f) PincherX 150 by Trossen Robotics.

Figure C-1: An overview of manipulator options.

C-2 Options

A list of options is created, first I inquired about manipulators present in Mechanical, Maritime and Materials Engineering (3mE) which can be taken home. Furthermore, a list is created of manipulators with MoveIt! support below the €1000,00 price tag. Useful sources were ROS Robots, MYBOTSHOP, Conrad and ROS COMPONENTS.

Ed-Ro

The Ed-Ro robot shown in Figure C-1a has 5 degrees of freedom which are actuated by a voltage-controlled DC motor and the angle of the joint is measured by a potentiometer. Analog signals are converted to digital and vice versa, in order to allow the control of the robot by means of a computer. It has an old RS232 interface and has proven to work with MATLAB/Simulink. No MoveIt! support is available, and except for two publications no information can be found online. The parallel gripper does not suitable for the task, however it might be modified or replaced by 3D printed fingers.

RAMrobot

The RAMrobot shown in Figure C-1b is a very simple 3DoF robot without an end-effector.

uArm

The uArm Swift shown in C-1c is developed for educational purposes. It supports programming via Arduino, Python, GRABCAD and ROS. It is available with several different end effectors among which a simple gripper. It can handle payloads up to 500g. The four degrees of freedom allow to control the pitch, but not the roll of the end effector. It is available via CONRAD or for €759.00.

OpenMANIPULATOR-X

The OpenMANIPULATOR-X shown in Figure C-1d is a open source robot platform. It has four degrees of freedom, but the end-effector roll can not be controlled. It can lift object of up to 500g. Since the STL-files are available, a custom end-effector can easily be designed and 3D printed. It can controlled via MoveIt!. It is available via MYBOTSHOP for €930.00 (including the base plate).

PhantomX Reactor

The PhantomX Reactor, as shown in Figure C-1e, is an older model designed by Trossen-Robotics. It was designed with entry-level research and university use in mind. A version with wrist joint is available. Object up to 150g can be lifted, limited by the wrist joint. It has ROS (indigo) support. Available via MYBOTSHOP for €749.00.

Name	Ed-Ro	RAMrobot	uArm	Open-X	PhantomX	PincherX 150
DoF	5	3	4	4	5	5
Rolling wrist	Yes	No	No	No	Yes	Yes
ROS Support	No	No	Yes	Yes	Yes*	Yes
Payload (g)	?	?	500	500	150	50
Availability	TU Delft	TU Delft	Netherlands	Germany	EU	Germany
Price (€)	0.00	0.00	759.00	930.00	749.00	1069.00

Table C-1: Overview of provided options. *Only support ROS Indigo

PincherX 150

The PincherX 150 is a newer model provided by Trossen Robotics and features the DYNAMIXEL X-Series Smart Servo Motors. It is constructed from extruded aluminium and aluminium brackets as shown in Figure C-1f. It comes with an wrist joint. The end effector can easily be exchanged for a custom design. It comes with ROS packages and support. Available via MYROBOTSHOP for €1,069.00. Unfortunately it is not recommended to lift objects over 50g. Stronger, but also more expensive versions, can be ordered from the US.

C-3 Discussion and Conclusion

An overview of the options if given in Table C-1. The most expensive solution, the pincherX, seems ideal. The PhantomX offers a more affordable alternative, but having to rely on ROS indigo may give some issues with the currently developed software. An alternative might be modifying the package to make it work with Kinetic. The uArm and OpenMANIPULATOR-X are not an option due to the missing rolling wrist joint.

Bibliography

[1] E. Heuvelink, *Tomatoes*, vol. 27. CABI, 2018.

[2] CBS, "Vooral tomaten in de kas," 2017. [Online; accessed 01-09-2020].

[3] A. N. Persbureau, "Trostomaat meest geteelde groente in glastuinbouw," 2017. [Online; accessed 18-03-2020].

[4] M. LTD, "Tomato industry," 2020. [Online; accessed 28-08-2020].

[5] M. Bergerman, J. Billingsley, J. Reid, and E. van Henten, "Robotics in agriculture and forestry," in *Springer handbook of robotics*, pp. 1463–1492, Springer, 2016.

[6] J. A. Foley, N. Ramankutty, K. A. Brauman, E. S. Cassidy, J. S. Gerber, M. Johnston, N. D. Mueller, C. O'Connell, D. K. Ray, P. C. West, *et al.*, "Solutions for a cultivated planet," *Nature*, vol. 478, no. 7369, p. 337, 2011.

[7] F. Rodríguez, J. Moreno, J. Sánchez, and M. Berenguel, "Grasping in agriculture: State-of-the-art and main characteristics," in *Grasping in Robotics*, pp. 385–409, Springer, 2013.

[8] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.

[9] H. Lin, F. Guo, F. Wang, and Y.-B. Jia, "Picking up soft 3D objects with two fingers," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3656–3661, IEEE, 2014.

[10] C. W. Bac, E. J. van Henten, J. Hemming, and Y. Edan, "Harvesting robots for high-value crops: State-of-the-art review and challenges ahead," *Journal of Field Robotics*, vol. 31, no. 6, pp. 888–911, 2014.

[11] E. Pekkeriet, "Picknpack; flexible robotic systems for automated adaptive packaging of fresh and processed food products," tech. rep., European Commission, 2016.

[12] C. Ji, J. Zhang, T. Yuan, and W. Li, "Research on key technology of truss tomato harvesting robot in greenhouse," in *Applied Mechanics and Materials*, vol. 442, pp. 480–486, Trans Tech Publ, 2014.

[13] N. Kondo, K. Yamamoto, H. Shimizu, K. Yata, M. Kurita, T. Shiigi, M. Monta, and T. Nishizu, "A machine vision system for tomato cluster harvesting robot," *Engineering in Agriculture, Environment and Food*, vol. 2, no. 2, pp. 60–65, 2009.

[14] W. Kuperberg, "Problems on polytopes and convex sets," in *DIMACS Workshop on polytopes*, pp. 584–589, 1990.

[15] A. Varava, D. Kragic, and F. T. Pokorny, "Caging grasps of rigid and partially deformable 3-d objects with double fork and neck features," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1479–1497, 2016.

[16] Y. Maeda, N. Kodera, and T. Egawa, "Caging-based grasping by a robot hand with rigid and soft parts," in *2012 IEEE international conference on robotics and automation*, pp. 5150–5155, IEEE, 2012.

[17] T. Egawa, Y. Maeda, and H. Tsuruga, "Two-and three-dimensional caging-based grasping of objects of various shapes with circular robots and multi-fingered hands," in *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*, pp. 000643–000648, IEEE, 2015.

[18] D. Kim, Y. Maeda, and S. Komiyama, "Caging-based grasping of deformable objects for geometry-based robotic manipulation," *ROBOMECH Journal*, vol. 6, no. 1, p. 3, 2019.

[19] Q. Feng, W. Zou, P. Fan, C. Zhang, and X. Wang, "Design and test of robotic harvesting system for cherry tomato," *International Journal of Agricultural and Biological Engineering*, vol. 11, no. 1, pp. 96–100, 2018.

[20] W. Lili, Z. Bo, F. Jinwei, H. Xiaoan, W. Shu, L. Yashuo, Q. Zhou, and W. Chongfeng, "Development of a tomato harvesting robot used in greenhouse," *International Journal of Agricultural and Biological Engineering*, vol. 10, no. 4, pp. 140–149, 2017.

[21] Q. Feng, X. Wang, G. Wang, and Z. Li, "Design and test of tomatoes harvesting robot," in *2015 IEEE International Conference on Information and Automation*, pp. 949–952, IEEE, 2015.

[22] M. H. Malik, T. Zhang, H. Li, M. Zhang, S. Shabbir, and A. Saeed, "Mature tomato fruit detection algorithm based on improved hsv and watershed algorithm," *IFAC-PapersOnLine*, vol. 51, no. 17, pp. 431–436, 2018.

[23] Y. Zhao, L. Gong, Y. Huang, and C. Liu, "Robust tomato recognition for robotic harvesting using feature images fusion," *Sensors*, vol. 16, no. 2, p. 173, 2016.

[24] Y. Zhao, L. Gong, B. Zhou, Y. Huang, and C. Liu, "Detecting tomatoes in greenhouse scenes by combining adaboost classifier and colour analysis," *biosystems engineering*, vol. 148, pp. 127–137, 2016.

[25] G. Lin, Y. Tang, X. Zou, J. Cheng, and J. Xiong, "Fruit detection in natural environment using partial shape matching and probabilistic hough transform," *Precision Agriculture*, vol. 21, no. 1, pp. 160–177, 2020.

[26] T. Yuan, L. Lv, F. Zhang, J. Fu, J. Gao, J. Zhang, W. Li, C. Zhang, and W. Zhang, "Robust cherry tomatoes detection algorithm in greenhouse scene based on ssd," *Agriculture*, vol. 10, no. 5, p. 160, 2020.

[27] G. Liu, J. C. Nouaze, P. L. Touko Mbouembe, and J. H. Kim, "Yolo-tomato: A robust algorithm for tomato detection based on yolov3," *Sensors*, vol. 20, no. 7, p. 2145, 2020.

[28] T. Yoshida, T. Fukao, and T. Hasegawa, "A tomato recognition method for harvesting with robots using point clouds," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, pp. 456–461, IEEE, 2019.

[29] M. Benavides, M. Cantón-Garbín, J. Sánchez-Molina, and F. Rodríguez, "Automatic tomato and peduncle location system based on computer vision for use in robotized harvesting," *Applied Sciences*, vol. 10, no. 17, p. 5887, 2020.

[30] E. Pekkeriet, "D5. 3 report on robot performance and lab test results," tech. rep., EU, 2015.

[31] I. Nyalala, C. Okinda, L. Nyalala, N. Makange, Q. Chao, L. Chao, K. Yousaf, and K. Chen, "Tomato volume and mass estimation using computer vision and machine learning algorithms: Cherry tomato model," *Journal of Food Engineering*, vol. 263, pp. 288–298, 2019.

[32] X. Li, Z. Pan, S. Upadhyaya, G. Atungulu, and M. Delwiche, "Three-dimensional geometric modeling of processing tomatoes," *Transactions of the ASABE*, vol. 54, no. 6, pp. 2287–2296, 2011.

[33] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[34] L. Busin, N. Vandenbroucke, and L. Macaire, "Color spaces and image segmentation," *Advances in imaging and electron physics*, vol. 151, no. 1, p. 1, 2008.

[35] I. E. Commission *et al.*, "Multimedia systems and equipment-colour measurement and management-part 2-1: Colour management-default rgb colour space-srgb," *IEC 61966-2-1*, 1999.

[36] S. CIE, "014-2/e: 2006: Joint iso/cie standard: ColorimetryâĂŤpart 2: Cie standard illuminants for colorimetry," *Vienna: CIE Bureau*, 2007.

[37] D. Zhang, M. M. Islam, and G. Lu, "A review on automatic image annotation techniques," *Pattern Recognition*, vol. 45, no. 1, pp. 346–362, 2012.

[38] H. Yuen, J. Princen, J. Illingworth, and J. Kittler, "Comparative study of hough transform methods for circle finding.," *Image and vision computing*, vol. 8, no. 1, pp. 71–77, 1990.

[39] T. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[40] J. Nunez-Iglesias, A. J. Blanch, O. Looker, M. W. Dixon, and L. Tilley, "A new python library to analyse skeleton images confirms malaria parasite remodelling of the red blood cell membrane skeleton," *PeerJ*, vol. 6, p. e4312, 2018.

[41] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[42] Y. Yu, K. Zhang, L. Yang, and D. Zhang, "Fruit detection for strawberry harvesting robot in non-structural environment based on mask-rcnn," *Computers and Electronics in Agriculture*, vol. 163, p. 104846, 2019.

[43] D. Ward, P. Moghadam, and N. Hudson, "Deep leaf segmentation using synthetic data," *arXiv preprint arXiv:1807.10931*, 2018.

[44] W. Shi, R. van de Zedde, H. Jiang, and G. Kootstra, "Plant-part segmentation using deep learning and multi-view vision," *Biosystems Engineering*, vol. 187, pp. 81–95, 2019.

[45] L. Luo, Y. Tang, Q. Lu, X. Chen, P. Zhang, and X. Zou, "A vision methodology for harvesting robot to detect cutting points on peduncles of double overlapping grape clusters in a vineyard," *Computers in Industry*, vol. 99, pp. 130–139, 2018.

[46] G. Lin, Y. Tang, X. Zou, J. Xiong, and Y. Fang, "Color-, depth-, and shape-based 3d fruit detection," *Precision Agriculture*, vol. 21, no. 1, pp. 1–17, 2020.

[47] T. Robotics, "Pincherx 150 robot arm." [Online; accessed 26-06-2020].

[48] A. Grunnet-Jepsen and D. Tong, "Depth post-processing for intel® realsenseâĎć d400 depth cameras," *New Technologies Group, Intel Corporation*, 2018.

[49] R. Y. Tsai, R. K. Lenz, *et al.*, "A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration," *IEEE Transactions on robotics and automation*, vol. 5, no. 3, pp. 345–358, 1989.

[50] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[51] K. Wada, "labelme: Image Polygonal Annotation with Python." https://github.com/wkentaro/labelme, 2016.

[52] W. Susanto, R. Babuška, F. Liefhebber, and T. van der Weiden, "Adaptive friction compensation: application to a robotic manipulator," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 2020–2024, 2008.

[53] R. Pérez-Dattari, C. Celemin, J. Ruiz-del Solar, and J. Kober, "Continuous control for high-dimensional state spaces: An interactive learning approach," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7611–7617, IEEE, 2019.