2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Occlusion Handling by Pushing for Enhanced Fruit Detection

Ege Gursoy[1,2] Dana Kulić[2] Andrea Cherubini[1]

***Abstract*— In agricultural robotics, effective observation and localization of fruits present challenges due to occlusions caused by other parts of the tree, such as branches and leaves. These occlusions can result in false fruit localization or impede the robot from picking the fruit. The objective of this work is to push away branches that block the fruit's view to increase their visibility. Our setup consists of an RGB-D camera and a robot arm. First, we detect the occluded fruit in the RGB image and estimate its occluded part via a deep learning generative model in the depth space. The direction to push to clear the occlusions is determined using classic image processing techniques. We then introduce a 3D extension of the 2D Hough transform to detect straight line segments in the point cloud. This extension helps detect tree branches and identify the one mainly responsible for the occlusion. Finally, we clear the occlusion by pushing the branch with the robot arm. Our method uses a combination of deep learning for fruit appearance estimation, classic image processing for push direction determination, and 3D Hough transform for branch detection. We validate our perception methods through real data under different lighting conditions and various types of fruits (i.e. apple, lemon, orange), achieving improved visibility and successful occlusion clearance. We demonstrate the practical application of our approach through a real robot branch pushing demonstration.**

Fig. 1: The robot arm pushes an occluding branch to improve the view of an orange.

I. INTRODUCTION

In natural environments, fruits are often occluded due to the complex and dynamic nature of their surroundings. Occlusions occur when other parts of the tree, such as branches, leaves, or even neighboring fruits, cover the fruit. This phenomenon is prevalent in dense foliage where overlapping branches and leaves create complex obstructed sightlines. Occlusions lead to incomplete or inaccurate observations, causing inaccurate detection and localization results, and therefore decreasing the reliability of a fruit-picking robot. In general, all agricultural robot applications which rely on visual sensors (e.g., fruit harvesting, yield estimation, disease detection and phenotyping) would greatly benefit from more precise observations. In the literature of fruit detection and occlusion handling, existing approaches often focus on estimating the entire fruit's properties from partially occluded views. While these methods provide valuable insight, they inherently rely on estimations that may introduce uncertainties and inaccuracies.

We propose a novel approach that goes beyond mere estimation. Rather than only relying on the prediction to find the appearance and location of occluded fruits, we take an active step, by physically clearing the occlusions which obstruct the fruit from the robot view. Our setup consists of an RGB-D camera, for identifying occluded fruits and occluding branches, and a robot arm to clear the latter. This approach enables us to obtain more accurate fruit observations, thereby improving the precision and reliability of fruit detection in complex agricultural environments.

To handle the occlusion by pushing the branch, we propose three steps: fruit identification, occlusion identification, and occlusion clearance. For fruit identification, we start by segmenting the visible section of the fruit in HSV color space, and then we apply a deep generative network to estimate the complete fruit in depth space. To identify occlusions, we design a guiding direction towards the occluding branch. Following this, we utilize HSV color space to segment branches attached to the tree. Then, we fit 3D line segments on the branches, via a 3D extension of the 2D Hough transform [1]. To clear the occlusion, we analyze the branches within the view frustum of the fruit, and compare their orientation with the view towards the fruit. We select the branch closest to the view normal, which indicates the pushing direction. Then, we determine the push point farthest from the fruit within the view frustum to avoid robot self-occlusions. Finally, we control the robot to push away the occluding branch.

The main contributions of this work are:

- we introduce a novel approach to handle occlusions in agricultural applications, by pushing branches away from the fruit view;
- we estimate the occluded parts of the fruit, by applying a deep generative network to the depth image;
- we extend Hough transform to 3D, to identify branches.

The rest of the paper is structured as follows. A review of the state of art in occlusion handling is given in Section II. Then, the problem statement and the overview of the method are explained in Sec. III. The fruit identification, occlusion identification and clearance are presented in Sections IV, V and VI, respectively. Finally, the results of the experiments are presented in Sec. VII, followed by a discussion and propositions for future work in Sec. VIII.

[1]LIRMM, Univ. Montpellier, CNRS, Montpellier, France
[2]Monash University, Clayton, Australia

II. RELATED WORK

Detecting and locating fruits is an important challenge in agricultural robots. Detection-only deep learning algorithms have gained significant traction among researchers in the domain. In our previous work [2], deep learning algorithms were used to detect and track oranges in images acquired by the robot RGB-D camera. Santos et al. [3] introduced an interactive instance mask generation method on RGB images, comparing the performance of various learning-based object detection algorithms on grape detection. Kang et al. [4] developed an automatic label generation module tailored for detecting apples in RGB images of orchards. Similarly, Koirala et al. [5] investigated popular object detection frameworks for mango detection, devising a custom detection network. Liu et al. [6] adopted a multi-class approach for citrus detection by categorizing samples into distinct classes. Kirk et al. [7] implemented a custom neural network that incorporated a 6-channel RGB + CIELAB representation as input for fruit detection tasks. Additionally, Ganesh et al. [8] trained a neural network using RGB and HSV 6-channel input for orange detection. Despite the effectiveness of these deep learning approaches, they are complex and not robust to occlusions. This emphasizes the ongoing need for adaptable detection methodologies.

Detection-only methods solely focus on improving the accuracy and efficiency of fruit detection, without addressing occlusions. Gao et al. [9] categorized apples based on occlusion conditions, to facilitate subsequent picking, while Fu et al. [10] categorized kiwifruits based on distance. Deep learning-based detectors have gained popularity, with approaches ranging from instance mask generation to custom neural networks, tailored for specific fruits.

Various approaches have been proposed to estimate occlusions on fruit. Feng et al. [11] developed a tomato-picking robot utilizing structured light and color thresholding. However, occlusion remained a significant cause of picking failures, as highlighted in Ling et al. [12] for a tomatoes use case. Kurtulmus et al. [13] employed statistical classifiers and neural networks to detect peaches, although challenges persisted with high occlusions. These methods indicate a growing focus on completing occluded fruits to enhance efficiency and reliability.

Branch detection methods also play a crucial role. Chen et al. [14] presented the dense composition of tomato plants, often occluded by branches, emphasizing the importance of accurate branch detection for effective fruit harvesting. Gong et al. [15] introduced a high-precision depth learning method for estimating occluded targets, while Lv et al. [16] developed a method for identifying occluded fruits in natural environments, utilizing dynamic threshold segmentation. These studies demonstrate the significance of branch detection, for overcoming occlusions in agrirobot applications.

In addition to technological solutions, Li et al. [17] experimented removing leaves beneath plants, to reduce target occlusion, while Gené-Mola et al. [18] applied airflow using sprayers to diminish occlusions. These labor intensive methods, demonstrate the potential of agronomic interventions in improving the visibility and accessibility of fruits.

The state-of-the-art primarily focuses on improving estimations. Our work distinguishes itself by prioritizing the clearing of occlusions, i.e. identifying the ground truth rather than relying solely on estimations.

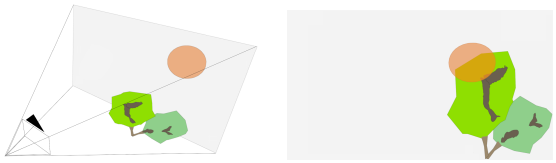

Fig. 2: Camera field of view (left), with corresponding image (right). The tree branches and leaves are green/brown, and the fruit is orange.

III. OUTLINE OF OUR WORK

A. Problem Statement

Consider a robot observing with a RGB-D camera a fruit tree. The goal of our work is to push, with the robot arm, the occlusions, which limit the visibility of the fruit. Fruits are partially observed by the robot because of the occlusions caused by other tree parts such as branches and leaves (see Figure 2). These occlusions lead to two significant problems. Firstly, the precision of fruit localization is compromised, since the robot can only perceive a partial view of the fruit. Second, creating a clear path for either the second arm or a person to access the fruit. Agricultural settings inherently contain numerous complex elements, and detectors may incorrectly identify fruits in shaded areas, leading to instances where a detected object is mistakenly identified as a fruit. These challenges emphasize the importance of addressing occlusions for more accurate and reliable fruit detection and localization.

We define two hypothesis in this work:

- *Hypothesis 1*: Undetected pixels of the fruit are in the camera field of view, reducing the image dimensions.
- *Hypothesis 2*: The fruits in this work are considered perfectly elliptical in shape, simplifying the complexity of the calculations.

B. Overview of our method

Our approach begins by segmenting the visible portion of the fruit using color information, to isolate the visible pixels of the fruit. These pixels are projected on the depth image, to obtain the corresponding 3D points. Then, we use a generative network to estimate the complete fruit in 2D. To this end, we only focus on the depth image, to reduce the network complexity. The network is trained using pairs of occluded/complete fruit depth images; it takes an occluded fruit depth image as input and generates an estimation of the complete fruit depth image. We determine the direction vector from the centroid of the estimated fruit to the centroid of the difference between the segmented and estimated fruit to calculate the direction needed to clear the occlusion (named *view gradient*). Further, we compute the imaginary line from the camera to the centroid of the estimated fruit, and we create the *view frustum* of the fruit around that line. We explore the point cloud of the environment generated by the RGB-D camera, and identify the points within the view frustum as the occlusion points. We search for 3D line segments in the point cloud and select the *pushing line*, based on their relationship with the view gradient and on their distance to the view frustum. Then, we define the *pushing point* depending on the location of the pushing line wrt the view frustum. The amount and direction of the push is set according to the view gradient and to the fruit size. Finally, we push the point with the robot arm, to clear the occlusion.

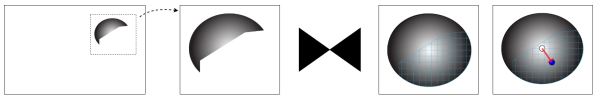

Fig. 3: Outline of the fruit estimation method. Left to right: Visible fruit pixels in masked depth image $\mathbf{D}_M$, cropped ROI depth image $\mathbf{D}'_M$ fed into generative model which outputs the estimated fruit depth image, center point of the fruit (white circle) $(\hat{u}'_c, \hat{v}'_c)$, center point of the $\bar{\mathbf{M}}'$ (blue circle) defined as $(\bar{u}'_c, \bar{v}'_c)$ shown on $\bar{\mathbf{D}}'_M$, occlusion direction vector $\vec{\mathbf{O}}_f$ (red arrow). Cross hatched area represents the area $\hat{\mathbf{D}}'_M$, estimated by the generative model.

IV. FRUIT IDENTIFICATION

A. Partial Fruit Segmentation

Let us denote $\mathbf{m}$, the mapping between pixel $(u, v) \in \mathcal{I} = \{1, \ldots, w\} \times \{1, \ldots, h\} \subset \mathcal{N}^2$ (from a camera of $w \times h$ resolution) and the 3D position of the corresponding point in the camera frame $\mathbf{P} = (x, y, z) \in \mathcal{F}_{cam} \subset \mathcal{R}^3$. We denote the projection of a pixel (u, v) in color information towards the depth information as $z_{uv} \in \mathcal{R}^+$. If we consider an undistorted perspective projection (i.e., a pinhole model), the mapping is:

$$\begin{array}{rccc} \mathbf{m}: & \mathcal{I} \times \mathcal{R}^+ & \rightarrow & \mathcal{F}_{cam} \\ & \begin{pmatrix} u \\ v \\ z_{uv} \end{pmatrix} & \mapsto & \begin{pmatrix} x = \frac{u-u_0}{f} z_{uv} \\ y = \frac{v-v_0}{f} z_{uv} \\ z = z_{uv} \end{pmatrix} \end{array} \quad (1)$$

with $(u_0, v_0) \in \mathcal{I}$ the camera principal point coordinates in the image plane and $f \in \mathcal{R}^+$, the camera focal length. Note that this mapping is *bijective*:

$$\begin{array}{rccc} \mathbf{m}^{-1}: & \mathcal{F}_{cam} & \rightarrow & \mathcal{I} \times \mathcal{R}^+ \\ & \begin{pmatrix} x \\ y \\ z \end{pmatrix} & \mapsto & \begin{pmatrix} u = xf/z_{uv} + u_0 \\ v = yf/z_{uv} + v_0 \\ z = z_{uv} \end{pmatrix} \end{array} \quad (2)$$

To find a fruit, we analyze the color image and specifically its Hue (H), Saturation (S), Value(V) histograms. We determine the H, S, V values corresponding to fruits and segment the corresponding pixels. We then retrieve the z values of these pixels from the depth, forming the depth image of the segmented fruit $\mathbf{D}_f$.

B. Fruit Appearance Estimation

From $\mathbf{D}_f$, we estimate the occluded part of the fruit using a generative neural network, based on an encoder-decoder architecture. Figure 3 outlines the method.

First, we apply a 2D binary mask $\mathbf{M} \in \mathcal{I} \times \{0, 1\}$ to $\mathbf{D}_f$. The resulting masked depth image is:

$$\mathbf{D}_M = \mathbf{M} \otimes \mathbf{D}_f \in \mathcal{I} \times \mathcal{R}^+, \quad (3)$$

with $\otimes$ as the Hadamard product. A visual representation of $\mathbf{D}_M$ is shown in the leftmost picture of Figure 3.

Under *Hypothesis 1* defined in Section III-A, the complete fruit should appear (if there were no occlusions) inside a rectangular region of interest (ROI).

We crop $\mathbf{D_M}$ to obtain the ROI $\mathbf{D}'_M$ by (4):

$$\begin{aligned} \mathbf{D}'_M = \{\mathbf{D}_M \mid u_{TL} \leq u_i \leq u_{BR}, v_{TL} \leq v_j \leq v_{BR}\} \\ \text{with} \quad (u_{TL}, v_{TL}) = w_s - d_{ROI}/2, h_s - d_{ROI}/2 \\ (u_{BR}, v_{BR}) = w_s + d_{ROI}/2, h_s + d_{ROI}/2 \end{aligned} \quad (4)$$

with w_s, h_s width and height of the segmented fruit and d_{ROI} ROI size, u_{TL}, v_{TL} and u_{BR}, v_{BR} as the top-left and bottom-right pixels of the ROI.

It is always possible to reinsert the ROI in the original image, by applying translation $\mathbf{r}$:

$$\begin{array}{rccc} \mathbf{r}: & \mathbf{D}'_M & \rightarrow & \mathbf{D}_M \\ & \begin{pmatrix} u \\ v \end{pmatrix} & \mapsto & \begin{pmatrix} u' = u + u_{TL} \\ v' = v + v_{TL} \end{pmatrix}. \end{array} \quad (5)$$

The network is trained with pairs of occluded and non-occluded $\mathbf{D}'_M$. Both the fruits and occlusions are generated at random locations. Non-occluded orange images are synthetic, created from ellipses colored similarly to the fruit, and occlusions are simulated by random croppings on the non-occluded images. The dataset consists of 100 images, divided 70:30 into training and validation sets. We feed the input with the occluded depth image and construct the loss by comparing the generated image and the non-occluded image.

The encoder-decoder network uses a modified U-net image generator from our previous work [19], with the following specifications. For the encoder, the size of the input layer is fixed to 56x56. The input is followed by 4 downscale layers, each consisting of two 3x3 convolution layers followed by a 2x2 maxpooling layer. Filter sizes of the convolution layers are 32, 64, 128 and 256, respectively. The decoder has 5 upscale layers and 1 output layer. Upscale layers consist of two 3x3 convolution layers, followed by a 2x2 upsampling layer. Filter size of convolution layers are 512, 256, 128, 64 and 32, respectively. Outputs of the second convolution layer from downscale layers are concatenated with the output of the symmetrical upsampling layer, before being fed to the following upscale layers. The output layer is a 1x1 convolution layer with sigmoid activation function. All the strides are set to 1 and rectified linear unit activation is used in all the convolution layers, except for the output layer. We use the binary cross entropy loss function in our network.

After training, giving an occluded $\mathbf{D}'_M$ as input, we can generate predicted image $\hat{\mathbf{D}}'_\mathbf{M}$ of the completed ROI. We can obtain the centroid pixels $\hat{u}'_c$, $\hat{u}'_c$ in $\hat{\mathbf{D}}'_M$ via:

$$\hat{u}'_c, \hat{v}'_c = \frac{1}{N_{\neq 0}} \sum_{i,j|z_{i,j} \neq 0} u_{i,j}, \quad \frac{1}{N_{\neq 0}} \sum_{i,j|z_{i,j} \neq 0} v_{i,j} \quad (6)$$

with $N_{\neq 0}$ the number of fruit pixels with $z_{i,j} \neq 0$ and $u_{i,j}$, $v_{i,j}$ their pixel coordinates. We also compute $\hat{w}_f, \hat{h}_f$, i.e., the width and height of the estimated fruit by subtracting maximum and minimum pixels with $z_{i,j} \neq 0$.

Finally, we calculate the projection of $(\hat{u}'_c, \hat{v}'_c)$ in $\mathcal{F}_{cam}$:

$$\hat{\mathbf{p}}_c = \begin{pmatrix} \hat{x}_c \\ \hat{y}_c \\ \hat{z}_c \end{pmatrix} = \mathbf{m} \begin{pmatrix} \hat{u}_c \\ \hat{v}_c \\ z_{\hat{u}_c, \hat{v}_c} \end{pmatrix} \quad (7)$$

with $\hat{u}_c, \hat{v}_c = \mathbf{r}[(\hat{u}'_c, \hat{v}'_c)]$.

In the rest of the paper, we consider the properties of the real (occluded) fruit identical to those of its estimated (entire) counterpart; i.e., we assume that the real fruits' properties p_c, w_f, h_f, u_c, v_c are identical to estimated $\hat{p}_c, \hat{w}_f, \hat{h}_f, \hat{u}_c, \hat{v}_c$ and the two sets can be used interchangeably.

V. OCCLUSION IDENTIFICATION

A. View Gradient

We generate two binary mask images $\mathbf{M}'$ and $\hat{\mathbf{M}}'$ from $\mathbf{D}'_M$ and $\hat{\mathbf{D}}'_M$, wherein pixels are set to 1 if their values differ from 0. We define $\bar{\mathbf{M}}' = |\mathbf{M}' - \hat{\mathbf{M}}'|$. We compute $area(\bar{\mathbf{M}}')/area(\mathbf{M}')$. If this ratio is bigger than a threshold r_o, we consider that the fruit has an occlusion. In this case, we compute $(\bar{u}'_c, \bar{v}'_c)$ the centroid of $\bar{\mathbf{M}}'$ by adapting (6) and we calculate the occlusion pixels direction (i.e., unitary vector) $\vec{\mathbf{O}}_f \in \mathcal{R}^2$:

$$\vec{\mathbf{O}}_f = \frac{(\bar{u}'_c - \hat{u}'_c, \bar{v}'_c - \hat{v}'_c)^T}{\|\bar{u}'_c - \hat{u}'_c, \bar{v}'_c - \hat{v}'_c\|}. \tag{8}$$

This vector, named *view gradient*, determines the direction of the most missings, and is the red arrow shown in Figures 3 and 5.

B. Branch Line Segments

First, we compute the H, S, V histograms as in IV-A. Then, we determine the branches values, to segment the branch pixels. We binarize the image by applying a threshold: if $z > 0$, the pixel is set to 1; otherwise, it is set to 0. We apply a Hough transform, to find straight line segments [1] $\mathbf{l}_{2D}$ defined by their image end points: (u_{l1}, v_{l1}) and (u_{l2}, v_{l2}). The main advantages of the Hough transform over other line fitting methods are its simultaneous detection of all lines. We design an extension of the Hough transform in 3D space to fit line segments $\mathbf{l}_{branch}$ into those branch points.

To this end, we define a distance threshold ρ_{th} and an angle threshold θ_{th}. We compare all detected lines and remove similar lines if their Hough distances wrt ρ and θ are smaller than ρ_{th} and θ_{th}. For each line segment $\mathbf{l}_{2D}$, we determine the plane $\mathbf{\Pi}$ in $\mathcal{F}_{cam}$, passing through the camera center and the line segment. The endpoints of the line segment in the image are projected onto the 3D camera frame $\mathcal{F}_{cam}$ using the mapping $\mathbf{m}$, where their z coordinate is set to focal length f. Plane $\mathbf{\Pi}$ can be expressed as:

$$Ax + By + Cz = 0, \tag{9}$$

with $\mathbf{N}_\Pi = (A, B, C) = (u_{l1}, v_{l1}) \times (u_{l2}, v_{l2})$ its normal vector.

To extend the Hough transformation to 3D, we project the branch pixels into depth information and form $\mathbf{P}_b$, branch points using mapping $\mathbf{m}$. Points in $\mathbf{P}b$ that are closer than a distance threshold d_Π to the plane $\mathbf{\Pi}$ are filtered out, resulting in the formation of $\mathbf{P}_b^\Pi$. Then, we rotate the points in $\mathbf{P}_b^\Pi$ around the camera center point to align the plane $\mathbf{\Pi}$ with the camera optical axis to form ${}'\mathbf{P}_b^\Pi$. The rotation matrix $\mathbf{R}$ about an arbitrary axis by angle α is calculated by Rodrigues' formula (with $\mathcal{F}^z_{cam}$ denoting the z axis of $\mathcal{F}_{cam}$):

$$\begin{aligned} \mathbf{K} &= \mathcal{F}^z_{cam} \times \mathbf{N}_\Pi \\ \cos(\alpha) &= \frac{\mathcal{F}^z_{cam} \cdot \mathbf{N}_\Pi}{\mathcal{F}^z_{cam} \cdot \mathbf{N}_\Pi} \\ {}'\mathbf{P}_b^\Pi &= \mathbf{R}(\mathbf{K}, \alpha) \cdot \mathbf{P}_b^\Pi. \end{aligned} \tag{10}$$

We form a new 2D image from ${}'\mathbf{P}_b^\Pi$ and reapply Hough transform to find straight line segments and remove similar lines to get $\mathbf{l}'_2D$. Since the line detected by the last process is an intersection of two planes, the resulting line can be expressed as a 3D line: $\mathbf{l}'_3D = \mathbf{m}(\mathbf{l}'_2D)$. We apply the inverse of the applied rotation to finally obtain $\mathbf{l}_{3D} = \mathbf{R}(\mathbf{K}, -\alpha) \cdot \mathbf{l}'_{3D}$, the 3D line segment in $\mathcal{F}_{cam}$. We define $\mathbf{l}_b$, containing all the $\mathbf{l}_{3D}$ from all of the $\mathbf{l}_{2D}$.

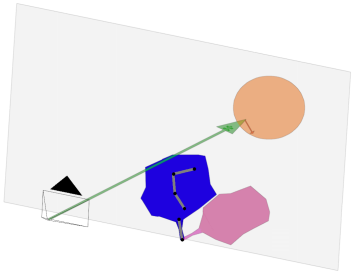

Fig. 4: Line segments (grey) and their extremities (black circles). The pushing line $\mathbf{l}_{push}$ $\mathbf{l}_{push}$ is selected as the topmost according to Algorithm 2.

All line finding steps are described in algorithm 1, where the image processing and line segment post-processing steps are shortened by *2D Hough line transform*.

Algorithm 1 Line finding algorithm

Input: Color and Depth information
Output: 3D branch line segments $\mathbf{l}_b$

```
Branch segmentation in HSV
Form an image from segmented pixels
2D Hough line transform on the image: l_2D
for l ∈ l_2D do
    Compute the plane Π
    Filter P_b by distance to Π: P_b^Π ← P_b < d_Π
    Rotation of the points P_b^Π: 'P_b^Π ← R(K, α) · P_b^Π
    Form a new image from 'P_b^Π:
    2D Hough transform on the new image: l'_2D
    3D mapping of l'_2D: l'_3D ← m(l'_2D)
    Inverse rotation of l'_3D: l_3D ← R(K, −α) · l'_3D
end for
return l_b which contains all the l_3D
```

VI. OCCLUSION CLEARANCE

A. Line Decision

The fruit frame $\mathcal{F}_{fruit}$ is defined as shown in Figure 5 with the z axis in the direction of the fruit:

$$\begin{aligned} \mathbf{f}: \quad \mathcal{F}_{cam} &\rightarrow \mathcal{F}_{fruit} \\ \begin{pmatrix} x \\ y \\ z \end{pmatrix} &\mapsto \mathbf{R}^{\mathbf{p}_c} \begin{pmatrix} x^\dagger \\ y^\dagger \\ z^\dagger \end{pmatrix} \end{aligned} \tag{11}$$

with $\mathbf{R}^{\mathbf{p}_c}$ the rotation matrix aligning $\mathcal{F}^z_{cam}$ to $\mathbf{p}_c$, which can be calculated as in (10).

Under *Hypothesis 2* defined in Section III-A, we compute the equation of the view frustum which connects camera and fruit, shown in Figure 5 and defined as:

$$\frac{x^{\dagger 2}}{\mathbf{m}(w_f)^2} + \frac{y^{\dagger 2}}{\mathbf{m}(h_f)^2} = \frac{z^{\dagger 2}}{||\mathbf{p}_c||^2}, \text{ s.t. } |z^\dagger| \leq ||\mathbf{p}_c|| \tag{12}$$

with $\mathbf{m}(w_f)^2$ and $\mathbf{m}(h_f)^2$ as the width and the height of the fruit in meters.

The branch to push is selected among the lines that are within a distance d_V to the view frustum V and have a

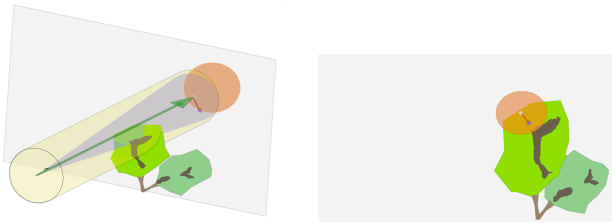

Fig. 5: Field of view from the camera (left) and Image (right). The figure shows: fruit centroid (white circle) $(\hat{u}_c, \hat{v}_c)$, occlusion centroid (blue circle) $(\bar{u}_c, \bar{v}_c)$, view frustum (blue cone), its center line $\mathcal{F}^z_{fruit}$ (green arrow) and view gradient $\vec{\mathbf{O}}_f$ (red arrow).

depth (z_{branch}) smaller than the fruit centroid depth (z_c). These points form $\mathbf{l}^{view}_{branch}$. Then, we project each line $\mathbf{l}_{3D}$ in $\mathbf{l}^{view}_{branch}$ to the image plane $\mathcal{I} \times \mathcal{R}^+$, via $\mathbf{m}^{-1}$. We compute the angle between $\mathbf{I}_{2D}$ and the view gradient $\vec{\mathbf{O}}_f$ as $\gamma = \alpha - \frac{\pi}{2}$; via γ, to identify the branch which is closer to the perpendicular to $\vec{\mathbf{O}}_f$. The line to push $\mathbf{l}_{push}$ is selected as the $\mathbf{l}_{3D}$ with the smallest γ. The pushing line decision algorithm is summarized in Algorithm 2. An example of Hough line transform output $\mathbf{l}^{view}_{branch}$ and the selected pushing line $\mathbf{l}_{push}$ are shown in Figure 4.

Algorithm 2 Pushing line decision algorithm

Input: Line segments $\mathbf{l}_{branch}$, Occlusion direction $\vec{\mathbf{O}}_f$
Output: Line to push $\mathbf{l}_{push}$

```
Filter branch segments: l^view_branch ← dist(l_branch, V) < d_V and z_branch < z_c
for l_3D ∈ l^view_branch do
    Project the l into image plane: l_2D ← m^{-1}(l_3D)
    Get the angle (l_2D, O_f) : α ← arccos [l_2D · O_f]
    Check α is how far from π/2: γ ← α − π/2
end for
return l_push ← l_3D with smallest γ
```

B. Push Action

If the pushing line $\mathbf{l}_{push}$ intersects with the view frustum, we select the initial pushing point among the points that intersect with the circumference of the view frustum V_C, opting for the one closer to the robot to minimize robot self-occlusions. Otherwise, we select the closest point to the view frustum, belonging to $\mathbf{l}_{push}$. We define the initial pushing point $\mathbf{p}_{push}$ as:

$$\mathbf{p}_{push} = \begin{cases} \mathbf{p} \cap V_C \text{ close to robot} & \text{if } \mathbf{l}_{push} \cap V = \emptyset \\ \arg\min \text{dist}(\mathbf{p}, V) & \text{otherwise} \end{cases} \quad (13)$$

with $\mathbf{p} \in \mathbf{l}_{push}$ and dist as the distance in 3D space and V the view frustum. Push magnitude is set to the largest distance between points in $\mathbf{M}'$:

$$d_{push} = \max_{\mathbf{p}_i, \mathbf{p}_j \in \mathbf{M}'} \|\mathbf{p}_i - \mathbf{p}_j\| \quad (14)$$

Finally, we move the robot end-effector from point $\mathbf{p}_{push}$ on a straight segment of length d_{push}, in the view gradient direction, $\vec{\mathbf{O}}_f$. The end-effector orientation is set to maintain a $\pi/4$ angle wrt the camera principal axis, and directed away from the image plane to prevent robot self-occlusions.

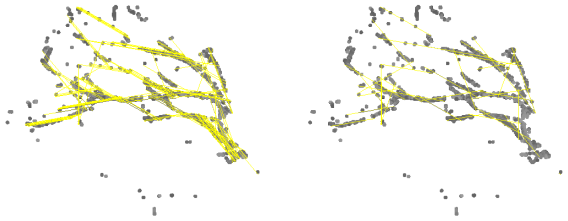

Fig. 6: Line detection before (left) and after (right) removing similar lines

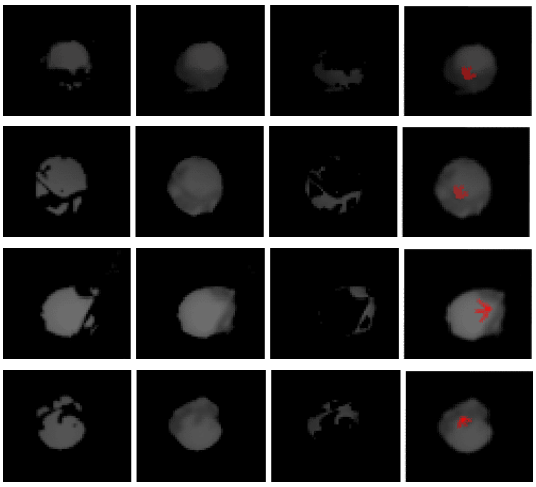

Fig. 7: From left to right: segmented fruit in depth image input to the fruit estimator, estimation result, difference between input and output, and view gradient (red vector) on the estimated fruit.

VII. EXPERIMENTS

Four experiments are conducted in three scenes under different light conditions and fruits (i.e. apple, lemon, orange) to test the perception algorithms. We run a case study with real robot experiment on an artificial orange tree. The video of the experiments attached to the paper is also available online at: youtu.be/watch?v=hGmj7sBb9vw.

We use the right arm (7-DOF KUKA LWR) of BAZAR robot [20] and an Intel Realsense D-435 RGB-D camera. A metal rod is attached at the center of the last arm link as a tool to perform the push action. We use Intel Realsense SDK for camera acquisition, and OpenCV for image processing. The robot is controlled using ROS Noetic with MoveIt.

The results of fruit appearance estimation, with the difference between the segmented fruit $\mathbf{M}'_M$ and its estimation $\bar{\mathbf{D}}'_M$ as well as the view gradient $\vec{\mathbf{O}}_f$ are shown in Figure 7. Fruits used in these experiments are: apple, orange, lemon, orange from top to bottom. The results suggest that while the shape of the fruit is accurately predicted, the depth estimation is less reliable. However, this is not crucial since we will binarize the image to obtain $\bar{\mathbf{M}}'$ from these predictions.

Different scenes are shown in Figure 8 with the branch and fruit segmentation results, including detected branch lines $\mathbf{l}_{branch}$, pushing line $\mathbf{l}_{push}$ and pushing point $\mathbf{p}_{push}$. The outcomes indicate that branches are generally accurately predicted, although there are instances of non-existent branch lines being detected. The selected pushing lines $\mathbf{l}_{push}$ and pushing points $\mathbf{p}_{push}$ appear reasonable from an empirical standpoint.

Figure 6 compares branch segmentation, before and after removing similar lines via Algorithm 1. Parameters used in

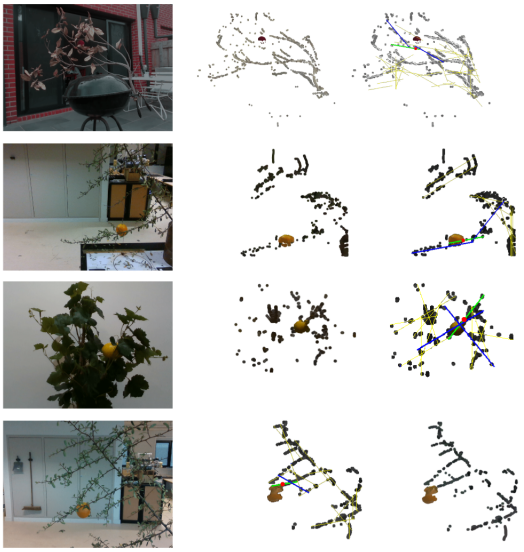

Fig. 8: From left to right: scene, segmentation results, and line detection results. For line detection: detected and filtered lines are yellow, lines below the distance threshold are blue, and the selected line is green. The pushing point is red.

Fig. 9: Robot branch pushing demonstration.

this example: $\rho_{th} = 25$ and $\theta_{th} = \frac{\pi}{6}$ rad. In other examples, ρ_{th} values vary between 15 - 25. The last column of Figure 8 is used as a case study to clear the occlusion with the robot. The robot in action is shown in Figure 9. The robot successfully clears the occlusion.

VIII. CONCLUSION

This paper presents a novel method for clearing fruit occlusions in agricultural applications. We begin by segmenting branches and fruit in the scene. We use a deep generative model to estimate the fruit's whole appearance. Then, we introduce a 3D extension of the 2D Hough transform, to detect branches as straight line segments. This enables us to identify the branch responsible for occlusion. We use the robot, to push the identified branch and clear the occlusion. We demonstrate the effectiveness of our perception methods under three different lighting conditions and across different types of elliptical fruit i.e. apple, orange, lemon. We present a case study involving a real robot demonstration.

While effective in controlled cases, our method faces challenges with branch segmentation and line detection, especially in complex scenarios and multiple occlusions. Scaling to multiple trees or integrating multiple robotic arms requires more research and robust real-time processing. Scalability involves adapting to different orchard types, tree densities, and environmental conditions, with customization needed for tree heights and shapes, and integration with existing machinery. Sensor fusion and adaptive machine learning can increase performance across varied settings.

Our approach of addressing occlusions by physically pushing obstacles can enhance detection precision by establishing ground truth instead of solely relying on estimations. In the future, this method can be generalized to different fruit shapes and integrate with multi-arm robots to create paths for fruit accessibility. We believe that the methods presented in this paper represent a step forward in agricultural robotics.

REFERENCES

[1] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[2] E. Gursoy, B. Navarro, A. Cosgun, D. Kulić, and A. Cherubini, "Towards vision-based dual arm robotic fruit harvesting," in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, 2023, pp. 1–6.

[3] T. T. Santos, L. L. de Souza, A. A. dos Santos, and S. Avila, "Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association," *Computers and Electronics in Agriculture*, vol. 170, p. 105247, 2020.

[4] H. Kang and C. Chen, "Fast implementation of real-time fruit detection in apple orchards using deep learning," *Computers and Electronics in Agriculture*, vol. 168, p. 105108, 2020.

[5] A. Koirala, K. Walsh, Z. Wang, and C. McCarthy, "Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'mangoyolo'," *Precision Agriculture*, vol. 20, no. 6, pp. 1107–1135, 2019.

[6] Y.-P. Liu, C.-H. Yang, H. Ling, S. Mabu, and T. Kuremoto, "A visual system of citrus picking robot using convolutional neural networks," in *2018 5th international conference on systems and informatics (ICSAI)*. IEEE, 2018, pp. 344–349.

[7] R. Kirk, G. Cielniak, and M. Mangan, "L* a* b* fruits: A rapid and robust outdoor fruit detection system combining bio-inspired features with one-stage deep learning networks," *Sensors*, vol. 20, no. 1, p. 275, 2020.

[8] P. Ganesh, K. Volle, T. Burks, and S. Mehta, "Deep orange: Mask r-cnn based orange detection and segmentation," *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 70–75, 2019.

[9] F. Gao, L. Fu, X. Zhang, Y. Majeed, R. Li, M. Karkee, and Q. Zhang, "Multi-class fruit-on-plant detection for apple in snap system using faster r-cnn," *Computers and Electronics in Agriculture*, vol. 176, p. 105634, 2020.

[10] L. Fu, Y. Feng, Y. Majeed, X. Zhang, J. Zhang, M. Karkee, and Q. Zhang, "Kiwifruit detection in field images using faster r-cnn with zfnet," *IFAC-PapersOnLine*, vol. 51, no. 17, pp. 45–50, 2018.

[11] 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*. IEEE, 2015, pp. 949–952.

[12] X. Ling, Y. Zhao, L. Gong, C. Liu, and T. Wang, "Dual-arm cooperation and implementing for robotic harvesting tomato using binocular vision," *Robotics and Autonomous Systems*, vol. 114, pp. 134–143, 2019.

[13] F. Kurtulmus, W. S. Lee, and A. Vardar, "Immature peach detection in colour images acquired in natural illumination conditions using statistical classifiers and neural network," *Precision agriculture*, vol. 15, pp. 57–79, 2014.

[14] X. Chen, K. Chaudhary, Y. Tanaka, K. Nagahama, H. Yaguchi, K. Okada, and M. Inaba, "Reasoning-based vision recognition for agricultural humanoid robot toward tomato harvesting," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 6487–6494.

[15] L. Gong, W. Wang, T. Wang, and C. Liu, "Robotic harvesting of the occluded fruits with a precise shape and position reconstruction approach," *Journal of Field Robotics*, vol. 39, no. 1, pp. 69–84, 2022.

[16] J. Lv, "Recognition of overlapping and occluded fruits in natural environment," *Journal of Computational and Theoretical Nanoscience*, vol. 13, no. 4, pp. 2475–2484, 2016.

[17] D. Li, X. Sun, S. Lv, H. Elkhouchlaa, Y. Jia, Z. Yao, P. Lin, H. Zhou, Z. Zhou, J. Shen, *et al.*, "A novel approach for the 3d localization of branch picking points based on deep learning applied to longan harvesting uavs," *Computers and Electronics in Agriculture*, vol. 199, p. 107191, 2022.

[18] J. Gené-Mola, E. Gregorio, F. A. Cheein, J. Guevara, J. Llorens, R. Sanz-Cortiella, A. Escolà, and J. R. Rosell-Polo, "Fruit detection, yield prediction and canopy geometric characterization using lidar with forced air flow," *Computers and Electronics in Agriculture*, vol. 168, p. 105121, 2020.

[19] E. Gursoy, S. Tarbouriech, and A. Cherubini, "Can robots mold soft plastic materials by shaping depth images?" *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3620–3635, 2023.

[20] A. Cherubini, R. Passama, B. Navarro, M. Sorour, A. Khelloufi, O. Mazhar, S. Tarbouriech, J. Zhu, O. Tempier, A. Crosnier, *et al.*, "A collaborative robot for the factory of the future: Bazar," *The International Journal of Advanced Manufacturing Technology*, vol. 105, no. 9, pp. 3643–3659, 2019.