Rubber Tree Tapping Position Detection on TrunkCovered RGB-D Images for Automation Platform

Rattachai Wongtanawijit[1] **and Thanate Khaorapapong**[2]

ABSTRACT: Recently, an autonomous vehicle with 3D sensors for the rubber tree (Hevea brasiliensis) plantation navigation has been presented. Therefore, we present a machine vision solution for detecting the tapping position and the rubber juice collecting cup in the images, which can be deployed onto an autonomous platform. First, we show an RGB-D image acquisition technique using artificial lights for capturing an image of rubber tree in the low-light. Then, we present two tapping position detection algorithms. One is a color-feature based with sliding window algorithm, and the other is a novel deep object detector. To perform the detection on our custom dataset, we built a FasterRCNN with the pre-trained MobileNetV2 using the finetuning technique. The results show that the deep detector outperforms our conventional detector and gives 92% average precision on our dataset.

Keywords: Object Detection, Hevea brasiliensis, Image Acquisition, Rubber Tapping, Computer Vision

DOI: 10.37936/ecti-cit.2021151.226743
Received November 26, 2019; revised March 20, 2020; accepted June 6, 2020; available online December 12, 2020

1. INTRODUCTION

Tapping for a rubber tree's juice, or "fresh latex", is always done by hand. Farmers use their traditional knives, "Je-bong", to tap the rubber trees. In tapping days, farmers tap all rubber trees in a plantation manually working in a low-light environment. In Southern Thailand [1], local farmers usually start tapping in the early morning for enhancing the latex yield. These labor-intensive tasks and dark working environments motivated us to develop automation for replacing human workers and improving the traditional farming processes.

Today, lots of robot applications use cameras to locate objects in the robot's workspaces for target manipulation. RGB-Depth cameras with compact sizes are more preferred than other sensors because they are more affordable with many advantages nowadays, such as effective operating power, and providing 2D or 3D images as the output. Many developments of robot vision show the use of cameras in ways that the acquired images can be simply processed by the algorithm to detect the objects with a small amount of time on mobile platforms where computation powers are considerably limited. We present a camera with a light positioning technique for capturing rubber tree images which is similar to Zhang's autonomous vehicle navigation pattern [2] and the automatic rubber latex farming robot patents such as [3], [4]. We chose the Intel RealSense camera D400 series [5] (D415 model) because it meets the requirements for Zhang's autonomous vehicle system. When making a rubber tree image dataset, we collect the tapping position RGB-D images from the rubber tree plantation during the actual farming period.

We collected 500 RGB-D images of highly cultivated RRIT-251 and RRIM-600 rubber trees during tapping process periods. We created tapping position ground truth data using bounding box annotations in all color images, which also affect the corresponding depth images with our bounding box placement regulations.

In our previous work [6], we showed the recognition of the tapping line-contained region using the tree trunk's color as the feature for the segmentation. This is used together with a modified histogram of oriented gradient (HOG) for extracting the tapping line's shape. This combination can be used to classify image regions to find tapping paths up to 65% of

[1,2]Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University, Thailand.
E-mail: rattachai.gov@gmail.com and kthanate@coe.psu.ac.th

the time, but the method still lacks the mechanism to generate a bounding box that relates to our ground truth.

So, in this project, we used a sliding window mechanism that replaces the modified HOG feature by computing color-segmented regions that are inside each window while limiting the window's sizes and the aspect ratios to fit the tapping path of the traditional spiral downward tapping system [7].

For the deep detector, we preferred a novel Faster-RCNN [8] for the tapping position detection because of the outstanding results on small objects compared to other deep-learning detectors and because of the size of the tapping position in the images. Our rubber tree image dataset is relatively small in size and very task-specific when compared to other datasets that are widely used to develop deep detectors such as ImageNet [9] or MSCOCO [10]. We implemented a fine-tuning technique to build Faster-RCNN detectors. We use the novel pretrained MobileNetV2 [11] as the backbone network for Faster-RCNN because it has a very small size and requires only modest computational power, which makes it feasible to use for a mobile platform. Since an RGB-D image is a 4-channel image that consists of red, green, blue, and depth grayscale-encoded components, an experiment that combined 4 channels into 3 channel images was established to build the Faster-RCNN detectors separately.

The results show that utilizing the depth image for tree trunk segmentation could help the detection precision which achieved 0.92 average precision (AP) at 0.5 Intersection-Over-Union (IoU). This is better than other algorithms that reach 0.33 AP at 0.75 IoU. Faster-RCNN is also better than our sliding window method that only produces 0.29 AP at 0.5 IoU for tapping line detection.

We first present a machine vision system using an RGB-D camera for rubber tapping automation. Second, we extend our previous color-based tapping line detection algorithm by adding a bounding box computation unit to enable the comparison to deep learning box detector which is measured by IoU and changes the detection algorithm type from data-supervised to a rule-based unsupervised algorithm. Third, we utilize the depth images to do a tree trunk segmentation in the preprocessing stage which increases the precision of the detection, thanks to the camera placement and the physical configuration of rubber tree plantation. Finally, we show the creation of deep learning detectors for detecting the tapping position which employ a novel Faster-RCNN detector with MobileNetV2 CNN (Convolutional Neural Network) architecture by using the fine-tuning technique on our dataset.

2. LITERATURE REVIEW

Related papers are reviewed in the order of vision systems of agricultural robotics, the traditional rubber tapping processes, the tapping path's appearance, rubber latex farming technologies, object detection algorithms for robotics applications, and deep learning object detection algorithms.

2.1 Machine Vision in Agricultural Robotics

Lots of today's agricultural robots emphasize the harvesting of the fruits from the trees. Examples include [12], [13] or in other forms of the crops such as [14], [15]. Their vision systems deploy traditional computer vision algorithms to recognize and locate the fruits' locations for grasping by the robotic arms. Since the robot's objective is to grasp the targets, the engineers who designed the machine vision system must calculate the grasping positions of the fruits by using object-specific detection algorithms coordinating with data from sensors. One example is target objects with circular shapes such as [16], [17]. Their algorithms computed the edges or shape's boundaries and other gripping factors such as object area or size to estimate target's weights, the center of mass for gripping points, etc. from the color images. Using prior facts that the algorithm designers knew about the object's physical structures and appearance, they deduced some mathematical models to solve the problem of searching for the target's boundary. The corresponding depth images were necessary for their algorithms to obtain 3D physical locations of the targets which can be calculated from stereo matching or other depth sensing techniques. But many of the mentioned algorithms only took the depth images as a lookup grid and they directly mapped the locations from algorithm-detected pixels on color images to the depth image find the depth distances from the target to the cameras.

Before the depth cameras were as affordable as they are nowadays, many vision systems used passive stereo methods to produce depth information from the scenes when required to obtain the target's 3D locations. The use of RGB-Depth cameras with active stereo sensing techniques has become much more popular in recent years due to lower prices, and as a result the developer community has grown. Compact size depth cameras which are sold by Intel [5] and Microsoft Kinect are used in object manipulation of agricultural robotics in [18], [19].

2.2 Tapping the Rubber Trees

Many farmers in Southern Thailand tapped the RRIT251 and RRIM-600 rubber trees using the downward spiral tapping system. They arrange the trunk area to tap, which is called the "tapping panel", and make the first tapping path 150 centimeters above the ground. On tapping days, farmers make a new tapping path adjacent to the most recent tap-

ping path which also expands the tapped area on the tapping panel down to the ground. When the tapping path reaches 50 centimeters in height, farmers will stop tapping on that tapping panel and start a new tapping on another tapping panel. The tapping path's length is between a quarter (denoted by S/4) or a half of the trunk's circumference (S/2) [20]. Their position lies from the farmers' higher left-hand to lower right-hand position with the average slopes at an angle between 30 to 45 degrees to level ground, but it could be tolerated by the farmers.

The traditional tapping processes begins when the farmers start tapping on a single tree by hand. Then they will walk to another tree to tap again, which means farmers have to travel to every tree in the rubber plantation. Also, the tapping times are usually between 3.00 to 6.00 am. in the morning. Farmers have to carry lights to help them do tapping, which is a hand-eye coordination task. After the farmers finish tapping all the rubber trees, they have to wait 1-2 hours to let the latex juice run into the collecting cup which is hung on the tree below the tapping path endpoint. Farmers collect the fresh latex from each cup by pouring the cup's latex into a bigger hand-held container. Thus, the duplicated task is that the farmers have to travel to all rubber trees in plantation again.

2.3 Rubber Latex Farming Technologies

Tapping machinery designs have been presented which utilize blade-like mechanisms with electrical motors. The designs replace the traditional knives and that lessens the need for highly skilled farmers. All parts are assembled into a hand-held tool (e.g. [21], [22]) that an unskilled farmer could grasp them to tap the rubber trees. The improved version is the tapping machine which must be attached to the tree's trunk [23]. These efforts require human workers to operate and can solve problems such as where to place the machines on all rubber trees, the powering of these machines, tolerances, and errors of tapping path creation due to rough surfaces on tapping panels, etc.

More practical automation systems of rubber latex farming have been presented such as [2], [24]. Especially the work of Zhang et. al [2], which showed an autonomous rubber tree navigation vehicle with 2D LiDAR and gyroscope sensors. The vehicle travels along the tree row and aligns itself to each tree with a robot arm at a tap-able distance. Thus, to develop a rubber tapping automation that utilizes an autonomous vehicle requires a vision system.

2.4 Object Detection Algorithms

Since the paper of R-CNN object detector (Region with Convolutional Neural Networks) was announced [25], many followers are showing some improvements over the original R-CNN in areas such as training times, computation efficiency, detection precisions, etc. FasterRCNN [26], YOLO [27], SSD [28] detectors have been developed which all outperformed the conventional handcrafted feature object detectors. As a result, these deeplearning detectors became popular for problem-solving in real-world large image datasets. Although the deep-CNN detectors perform well on many large datasets, the machine vision task of agricultural robotics is still not well explored by these algorithms because the data is too object-specific and platform dependent. Algorithm developers have to collect their data from actual working environments with their under-developed systems and build their deep detector using the fine-tuning method with pre-trained networks.

One difference between one-stage and two-stage deepCNN architectures is the computation unit to refine the predicted object locations in an image. They have different small object detection performance. For example, with two-stage CNN, a sub-network in Faster-RCNN shows that it can learn to predict the object locations using their alternated training techniques by being given the initial locations called "anchors". But one-stage CNNs employ a searching grid within a convolutional layer that behaves like the sliding window to locate objects within the grid. The small objects could lie within a cell or in-between adjacent cells of a large grid, causing the network fail to produce fine enough position output for those objects. The tapping position which we want to detect has a relatively small size compared to the whole image. This makes us consider using the two-stage detectors that perform well on small sized object detection.

The fine-tuning technique enables a solution that the developers can build for their deep-CNN detectors by adding new high-level convolutional layers or replacing the pre-trained high-level classification layers of the pretrained networks to match their problems. There are no exact rules or documentation that describe how to choose a pre-trained network for fine-tuning. [29] shows comparisons of the novel pre-trained CNNs in computational cost, classification accuracy and the sizes of parameters. MobileNetV2 has some advantages over other CNNs thanks to depth-wise convolution and the bottleneck architecture. We selected the pre-trained MobileNetV2 for our problem because we want to deploy a tapping position detection algorithm on a mobile computing platform.

The image from the RGB-D camera is in a 4-channel format – red, green, blue, and depth, but the pre-trained CNNs only take 3-channel images as input. Thus, we converted 4-channel images into 3-channel images and fed them into a CNN detector. Tapping position and harvesting cup detection performance was observed and analyzed for each combination.

Many of deep-CNN research efforts show that their detectors outperform many traditional detection al-

gorithms. In many object-specific applications, however, the developers still employ their own custom-designed detection algorithms because they can analyze the algorithm's behaviors with corresponding mathematical models and also tune all algorithm parameters by themselves without re-training new CNN detectors.

3. TAPPED RUBBER TREE IMAGES

Due to the high variation of natural rubber tree breeds, we limit our study for this research by only collecting images of RRIT-251 and RRIM-600 tapped rubber trees with ages between 8-10 years old from an area in the south of Thailand.

3.1 RGB-D Image Acquisition

We calculate the camera's position like the work of Zhang et. al [2] by using the Intel RealSense D415 camera as the image capturing device. The camera is placed between 90 to 120 centimeters from the tapping panel (camera's depth axis) and mounted on a tripod in portrait orientation at a height of 100 centimeters from the ground. This makes sure all sub-sensors inside the D415's field of view can cover the tapping panel at 50 to 150 centimeters above the ground. Two LED (Light Emitting Diodes) rods are positioned between the camera symmetrically with the same camera's depth as shown in Fig.1. These lights produce the scene with a natural look as in daytime which approximately has a 5500K color temperature. We adjust the scene's brightness to 80-150 lux measured at the trunk surfaces at the heights which are 50, 100, and 150 centimeters.

We keep the camera's settings at the provider's default presets for capturing color and depth images. The color image is in RGB color space and the depth image is in a single 8-bit grayscale format. We duplicate the depth image into 3-channel images which can be saved in a 24-bit PNG file format like a color image. The pixel values of the 8-bit grayscale image are in the integer range of 0-255. We mapped the camera's depths which are between 50 to 200 centimeters to be in pixel values in the 0-255 range by deciding that 255 represents 50-centimeter depth, and 0 represents the depths that are more than 200 centimeters from the camera depth plane. This is done to apply the depth images as the trunk's segmentation masks that can segment the rubber tree trunk by using the pixel-value thresholding method without using the corresponding color images.

We collected 500 sets of RGB-D images from 300 different tapped rubber trees. Since we want to create a tapped rubber tree dataset for tapping position detection, we force the camera to capture the images by limiting the tapping position appearances. The tapping path has the minimum width which equals half of the appearance of the trunk's width on the images.

Fig.1: *RGB-D Camera and Assistive Lights are turned to a Rubber Tree in low-light environment.*

3.2 Box Annotations for Tapping Positions

The rectangular box annotation, or "bounding box", can be described by 4 integers which are boxes' reference pixel positions and their size. The values are width and height, formatted in $\{u_b, v_b, w_b, h_b\}$. This box annotation is supported by deep learning object detectors (and also conventional window detectors) which can output 4 numbers for a box that contains a detected object.

Dataset makers usually draw the bounding boxes onto the images of objects by creating their drawing following regulations related to each object's appearance, such as aligning each side of the bounding box to fit the object's boundaries. That annotation method cannot be applied to the tapping positions because a tapping position or tapping path is a line or a curve which is a boundaryless instance. So, we place the box's top-left pixel (u_b, v_b) at the tapping path's most left pixel and the box's bottom-right pixel $(u_b + v_b, w_b + h_b)$ at the tapping path's right-position, as shown in **Fig.2** and **Fig.6**. This step is done on a color image and is also easily applied to the corresponding depth image automatically due to the fact that the color and depth images are already pixel aligned.

A tapping path is a curve with a consistent slope. When the tapping path is captured on the image, we can approximate a tapping path to be at the main diagonal of the annotated box that connects the box's top left point to the box's bottom right point. This rule is used for our camera placement and the downward tapping system because a tapping path appears to be a straight line in the captured image. This line is called the "tapping line". This annotation method differs from our previous work [6] because we annotated a tapping line by creating a bounding box around the tapped area region. The box annotations also make the tapping path invariant to the tapped region size or the expansion of the tapped area.

Fig.2: *Tapping Position and a Harvesting Cup annotationsin a ColorImage and its Corresponding Depth Image.*

4. THE COLOR-BASED DETECTOR

We updated our previous tapped region classifier by using the candidate bounding boxes with the detection scores as shown in **Fig.3**. In the previous work, the "*lower boundary*" (the bottom facing borders of the region) was highly affected by morphological erosion filters applied to the red color clustered images. The average slopes of the tapped region's lower boundary should follow the shape of the tapping path of the downward tapping system. We generate the candidate bounding boxes by fixing the box's aspect ratios related to the trunk's width so that their diagonal angles follow the tapping path slopes.

The algorithm does not require data for learning. All 500 sets of images are fed to the algorithm and the detection precision can be evaluated.

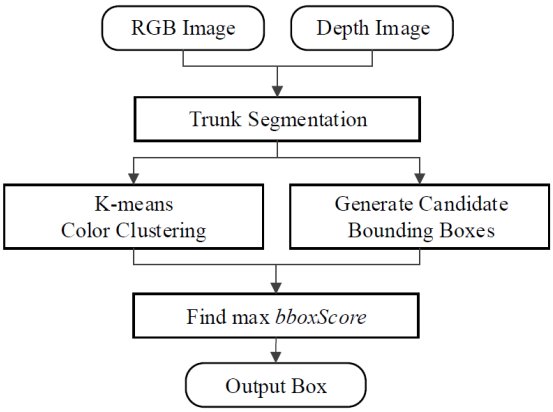

Fig.3: *Color-based Sliding Window Detector.*

4.1 Candidate Bounding Boxes

Let w_t denote the average trunk width of each image which can be computed by averaging the columns on the trunk mask image. Each candidate bounding box's width (w_b) is between $0.5w_t$ to $0.1w_t$. The box's height is related to tapping path's slopes which are between 30 to 45 degrees (θ_b). Since we approximate a tapping path using the main diagonal of the box, the height of the box is given by equation (1), as shown in Fig.4 This method controls the numbers of candidate boxes by limiting the sets of w_b and θ_b values.

$$h_b = w_b \tan \theta_b \tag{1}$$

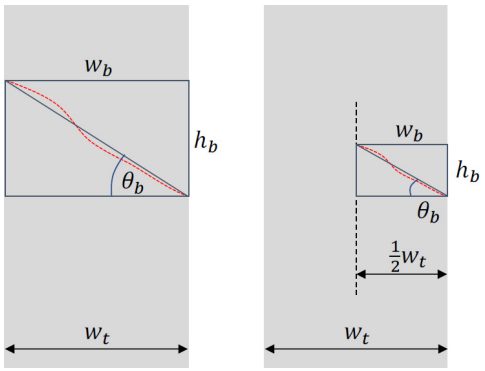

Fig.4: *Illustration of Candidate Bounding Boxes on Trunk Areas (Shaded) with Highest Width (Left) and Smallest Width (Right).*

4.2 Color Clustering for Tapped Area

We substitute the color clustering process of previous work, which performs on a hue channel image of HSV color space by the a*-channel, from CIELAB (L*a*b*) color space. We do this because by the eye's observation, a tapped region on the tapping panel is in red color tone and its neighboring regions are influenced by the natural appearance of the bark and other small vegetation whose color is greener than a tapped region. These properties of color tone are associated with the information of a* channel in CIELAB color space.

We initialize three k-means cluster centroids to $\{\infty, 0, -\infty\}$, which clusters the a*-channel pixels' values into 3 groups. After the k-means algorithm has converged, we choose the highest average group to be our feature map, which represents the high red color tone. This change reduces the number of clusters from our previous work that relied on hue values. The pixels which have unnecessary colors will not be processed in the k-means algorithm.

Like the previous work, the bottom facing border, which is named the "lower boundary" of the tapped region, or tapped area, gives a trace of the tapping path. We apply a box searching algorithm onto this clustered image (feature map) which finds the box from all candidate boxes that contains the lower boundary of the tapped region.

Fig.5: *A Red Color Tone Cluster Obtained from Kmeans Clustering on a Depth-masked a* Channel image.*

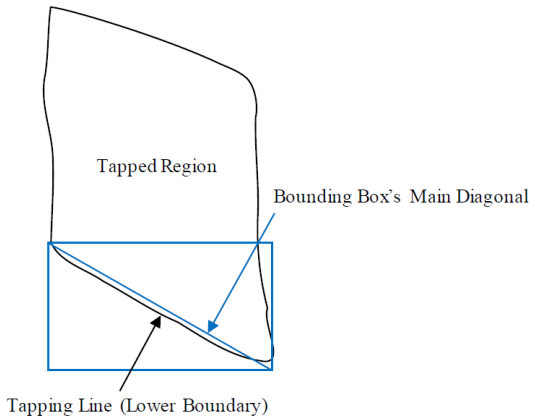

Fig.6: *A Tapping Line-Aligned Bounding Box.*

4.3 Detection with Sliding Windows

We define the scores for candidate bounding boxes with equation (2), which calculates the ratios of the area difference between upper and lower triangles to the box's area. Let nA_{upper}, nA_{lower} be the sum of valued pixels ("1" pixel values of the binary image) that lie above and below the bounding box's main diagonal respectively.

$$boxScore = \frac{nA_{upper} - nA_{lower}}{w_b \times h_b} \tag{2}$$

The *boxScore* ranges from -0.5 to 0.5, where the higher values mean that the box positions are at the tapped area-like regions. The $nA_{upper} - nA_{lower}$ term indicates the tapped region position and aligns the box's main diagonal to that region's lower boundary.

We compute *boxScore* for all candidate bounding boxes. The detection finds the box which has the highest score. But the computation of equation (2) encounters a scaling problem since the candidate boxes with smaller sizes could have a higher chance to gain higher scores because the small candidate boxes always cover smaller areas than the larger boxes, which means a small box has a lower area of variation. For example, if a small box is in the middle of the tapped region's lower boundary, its score might reach the maximum. Thus, we weight all the box scores by their areas so that larger boxes will be punished by their rank of size. Only one candidate box with the highest score is selected to be the output box in any particular image.

5. FASTER-RCNN DETECTOR

We built a faster-RCNN detector using the fine-tuning features of the MobileNetV2 by deploying faster-RCNN's layers and by creating high-level layers for RPN (Region Proposal Network) into an intermediate block of MobileNetV2. We added a new 2D convolution layer which takes the input tensors from "depthwise_relu" layer of the 10^{th} block of MobileNetV2 and a standard ReLU (Rectified Linear Unit) layer, then we connect the ReLU outputs to a classification layer (SoftMax) for object-ness prediction and also to a regression layer for ROIs prediction. Our annotated tapping position's bounding box has a small size, so we choose the 10^{th} block which has 14×14 spatial dimensions that are not too small or too large for RPN to locate the ROIs. Thus, the RPN is a subnetwork that is a part of the original MobileNetV2 from 1^{st} to 10^{th} blocks. The outputs of RPN are fed into the ROI pooling layer as documented in the Faster-RCNN paper [26]. Our detector is shown in **Fig.7** and **Fig.8**.

The output tensors from RPN are fed into the rest of MobileNetV2 starting at the "*depth projection*" layers of the 10^{th} block and continue the convolution through the last block of inverted residuals and linear bottleneck architecture of MobileNetV2. We added a class prediction layer and another bounding box regression layer for refining the ROIs to the detected objects at the top level.

In addition, we also added another output to the classification layer for the cup detection.

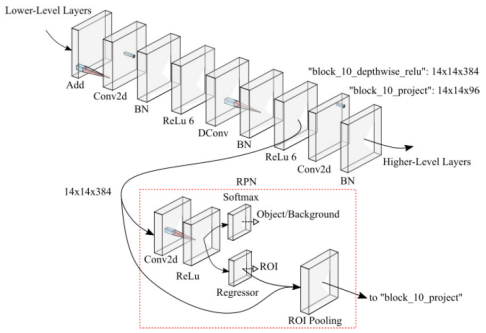

Fig.7: *Diagram Showing the High-Level Layers of RPN Built at 10^{th} Block of MobileNetV2.*

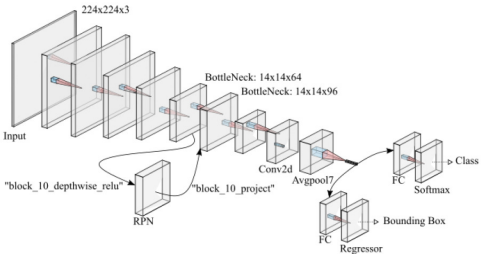

Fig.8: *Overview of Faster-RCNN with MobileNetV2.*

5.1 RPN Training

A faster-RCNN detector utilizes the anchors (initial bounding boxes) for training the RPN. The anchors are also defined by a set of 4 numbers that are fed into the ROI pooling layer during the training steps. The original paper [8] employs boxes with 3 different aspect ratios and scales. Then it selects the boxes which have the IoU ratios close to the ground truth boxes between 0.70-1.00 as the positive training samples, and 0.0-0.30 for the negative samples.

Their anchor initialization aims to cope with the detection of the generic objects that usually appear in realworld images. However, for the tapping position detection, their anchor definitions might cause a class imbalance problem — the region proposal network is trained to predict negatives with biased negative training data. Thus, we adjust the anchors' threshold by using 0.1-0.3 IoU for negatives, and 0.7-1.0 IoU for positives, which forces the negative samples to be around the ground truth boxes but not the clear background boxes. Also, for the anchors ratios and size, we choose the anchor's aspect ratios and sizes obtained from the ground truth data.

The number of region proposals for RPN training is fixed to be constant at 100 proposals per image and we sample them equally from positive and negative anchors.

5.2 Detector Network Training

The regular MobileNetV2 CNN takes a tensor size of 224× 224×3 (image's width, height, and channel or depth) for an input which usually is an RGB image. This network is also pre-trained with the RGB images of the ImageNet dataset in which all images are resized to match the network input's size. Unlike the usual image datasets, our tapped rubber tree images are in the 4-channel format, with RGB and depth, and sizes are 1280×768×4. The image resizing algorithm in the pre-processing stage of CNNs detector only copes with the width and height dimensions. A 4-channel image cannot be fed into the pretrained network directly. Thus, we conducted the experiments where we converted the 4 channels into a 3-channel image for training different faster-RCNN detectors to detect tapping and harvesting cup position.

We arranged the 3-channel images generated from 4-channel RGB-D images into 6 groups and used them to train and evaluate 6 different faster-RCNN detectors. The data groups were:

1. R/G/B: a standard 3-channel color image.
2. D/D/D: a duplicated-into-3-channel depth image.
3. Masked R/G/B: a color image which is masked by a trunk mask (binary image) obtained from depth image thresholding using pre-defined acquisition depth ranges—or depth segmentation.
4. R+G/B/D: merge R and G into one channel by alpha blending technique with equal ratios and two other channels are blue and depth.
5. R+B/G/D: blend R/B together as 4^{th} group.
6. G+B/R/D: blend G/B together as 4^{th} group.

To employ the fine-tuning technique to the faster-RCNN with MobileNetV2, we froze the weights (learnable parameters) of the low-level layers and allowed the newlyadded layers, or higher-level layers, to learn the training data, or allowed the lower layers to learn with small learning rates. We also used a k-fold cross-validation technique when k is 5 for training and evaluating all 6 of the detectors with 6 combined data groups. A standard SGDM optimizer (Stochastic Gradient Descent with momentum) was applied for the weight adjustments on faster-RCNN's loss functions during training.

6. RESULTS

The detection results from both algorithms were evaluated in 2 levels, using a traditional Pascal VOC's threshold at 0.5 IoU and a 0.75 IoU for the higher localization measurements as in [30].

6.1 Tapping Position Detection

For the color-based sliding window detector described in section 3, the number of candidate bounding boxes depends on various parameters which are θ_b, w_b, and the box's sliding or shifting distances

in the image's axis. To simplify the experiment, we examined a set of the fine values of such parameters which were, $w_b = \{.5, .6, .7, .8, .9, 1 \times w_t\}$, $\theta_b = \{27°, 30°, 37.5°, 45°, 48°\}$ (a downward tapping path with 3 degrees tolerance) and 5 pixels shifting distance in x and y axis. This examined parameter set produced a large number of candidate bounding boxes ($\approx$ 7-10k). These parameters made the searching become almost a brute force approach that should give the highest possible detection with practical computation.

The average precisions of tapping position detection were 0.291 and 0.029 when using the threshold of 0.5 IoU and 0.75 IoU respectively. These APs were used as the baseline values for the comparisons.

We also evaluated the detection precisions with varying IoU thresholds to observe the algorithm's characteristics, as shown in **Fig.9**. The algorithm produced the output boxes that gave reasonable positions for APs which ranged from 0.4 to 0.8 when using IoU thresholds less than 0.5 but they were not well-aligned to ground truth box when comparing them to faster-RCNNs. The result shows that the average precisions drop drastically when increasing the IoU thresholds.

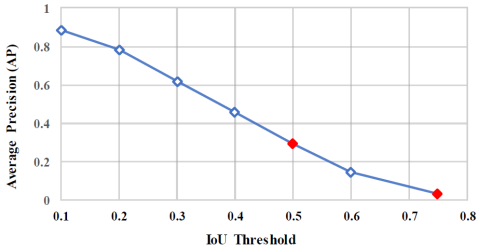

Fig.9: *Average Precisions of the Sliding Window Detector with Tapping Position Detection Evaluated Against Multiple IoU Thresholds.*

For the faster-RCNN detectors, the detection results were also measured using average precision metric on all detectors which were trained by all 3-channel combined data groups in section 5.2. We tested for AP by adjusting the number of proposals used for inferences. The tapping position detection APs evaluated at 0.5IoU and 0.75IoU are shown in **Fig.10** and **Fig.11**.

The number of RPN proposals also affects the detection precision because higher numbers of proposals may increase duplicate boxes around the tapping position, and they might be merged into one by a non-maximal suppression algorithm, which could lower the detection precision.

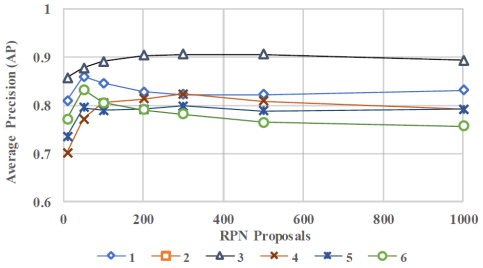

Fig.10: *Average Precisions of Faster-RCNN with Tapping Position Detection Evaluated at 0.5 IoU.*

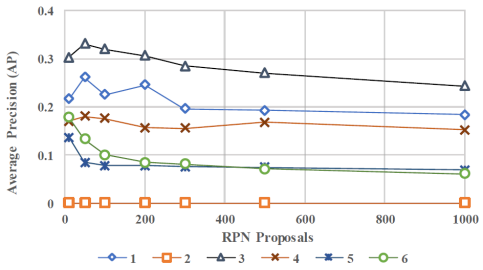

Fig.11: *Average Precisions of Faster-RCNN with Tapping Position Detection Evaluated at 0.75 IoU.*

6.2 Harvesting Cup Detection

In addition, we added the detection of the harvesting cup using the 6 faster-RCNN detectors. We added the harvesting cup classification output at top-level layers for fine-tuning. The cup detection results are shown in **Fig.12** and **Fig.13**, which were evaluated by 0.5 and 0.75IoU, in the same manner as in the tapping position detection.

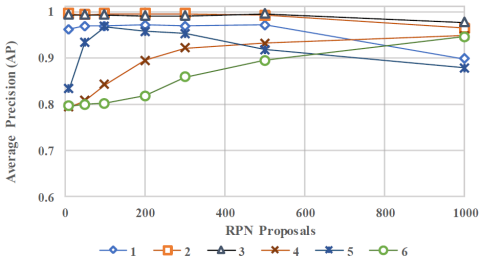

Fig.12: *Average Precisions of Faster-RCNN with Harvesting Cup Detection Evaluated at 0.5 IoU.*

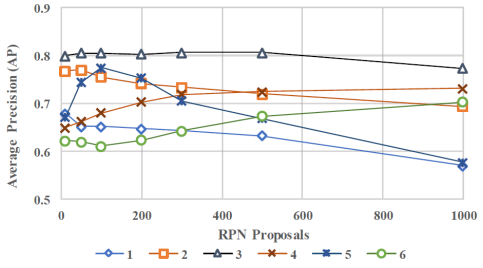

Fig.13: *Average Precisions of Faster-RCN with Harvesting Cup Detection Evaluated at 0.75 IoU.*

Fig.14: *Examples of the Faster-RCNN detection in RGB image (1^{st} group) and the Depth image (2^{nd} Group).*

6.3 Detector Performance Comparison

All fine-tuned Faster-RCNN with MobileNetV2 detectors outperform our conventional detector in average precision measurements which reach 0.920 against a 0.291 baseline at 0.5 IoU. This was especially true for a fasterRCNN detector that trained on the 3^{rd} data group which contains the RGB images that were masked by depth images. This high AP comes from the fact that the MobileNetV2 can extract more useful features than other groups of the input data. For the color with depth-masked data of the 3^{rd} data group, background variations are reduced by depth images which remove non-trunk area in the color images in contrast to other groups.

The experimental results also agree with our previous work which showed that a tapping path on the tapped area can be detected by the red color feature. Evidence proved that in the 4^{th}, 5^{th}, and 6^{th} groups, if we blend red or green channel (higher significant features) with the blue channel (fewer significant features), the detection precisions will be affected, causing lower detection precision, because a tapping position is always at the bottom of the tapped area which is red, connected to the green tone non-tapped area (natural bark). This is obvious in the high differences in **Fig.11** when using 0.75 IoU threshold.

Apart from harvesting cup detection, a black plastic bowl cannot easily be pointed out by the detection of only the RGB images compared to the masked RGB group. However, these cups can be detected using only depth information which is demonstrated on the detector of 2^{nd} group with also has high localization to other combinations as shown by the average precision at 0.75 IoU threshold in **Fig.13**.

The plot in **Fig. 15** shows the average precisions at multiple IoU thresholds of the color-based detector and the Faster-RCNN which is using depth for trunk segmentation in the pre-processing stage with 200 PRN proposals for inferencing. The results confirm that Faster-RCNN outperforms the color-based detector in all IoU thresholds.

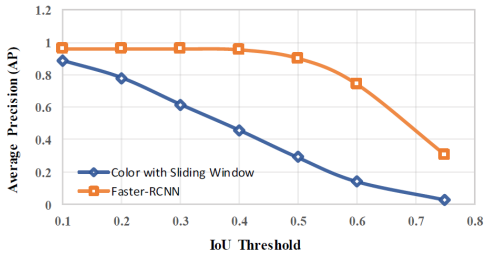

Fig.15: *Average Precision of Tapping Position Detection of Both Detectors Evaluated at multiple IoUs.*

7. CONCLUSIONS

To establish the tapping position detection algorithm for rubber tree farming automation platform, we proposed the use of an Intel RealSense RGB-D camera which can capture the color and depth images of tapped rubber trees. We show the camera positioning relative to the rubber tree which is needed by the autonomous rubber tree navigation robot [2] so that the camera's field of view can cover all of the tap-able area.

The development of the detection algorithms occurred in two phases. We began with a traditional box detection algorithm utilizing a sliding window and color clustering techniques. This method detects the tapping position's boxes in an unsupervised fashion. Our results show that using only color features obtained from the clustering of a* channel in CIELAB color space produced lower box localization than the deep-CNNs detector.

In the second phase, the deep learning detector was presented. We applied the fine-tuning technique to build the faster-RCNN detectors with the MobileNetV2 for our data. The deep-CNNs detectors show much higher detection precision than our unsupervised algorithm. Moreover, the tapping process is done within the nighttime. However, the camera requires artificial lighting to function, and it is necessary for tapping position detection which relies on color images. But for the harvesting cup detection, we showed successful deepCNNs detection, especially when using only depth images. That means the harvesting cup detection requires no additional lights. That is a contribution because tapping and harvesting periods occur in low-light times of the day

For further improvement, we propose tapping position box annotations that can be utilized for another near-range depth camera on the automation

platform. A near-range depth camera can acquire more details of a trunk's surface inside the annotated boxes, which is necessary for tapping path trajectory generation of the robotic arm.

References

[1] P. Thala, N. Kaewhgam, K. Kumnornaew and K. Satjawattana, "Effects of tapping time period and tapping system on latex yield of rubber trees (Hevea brasiliensis) at University of Phayao, Phayao, Thailand," *Khon Kaen Agriculture Journal*, Vol.42, pp.353-359, 2014.

[2] C. Zhang, L. Yong, Y. Chen, S. Zhang, L. Ge, S. Wang and W. Li, "A Rubber-Tapping Robot Forest Navigation and Information Collection System Based on 2D LiDAR and a Gyroscope," *Sensors 2019*, vol.19, no.9, pp.1-21, 2019.

[3] "Automatic integrated rubber tapping and collecting method based on image identification and automatic integrated rubber tapping and collecting device based on image identification," CN105494031A, 02-Feb-2016.

[4] "Rubber garden cable type automatic rubber cutting and harvesting device and method," CN108029500A, 07-Dec-2017.

[5] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen and A. Bhowmik, "Intel(R) RealSense(TM) Stereoscopic Depth Cameras," *I2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Honolulu, HI, no. 1, pp.1267–1276, 2017.

[6] R. Wongtanawijit and T. Kaorapapong, "Rubber Tapped Path Detection using K-means Color Segmentation and Distance to Boundary Feature," in *2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Chiang Rai, Thailand, pp.126–129, 2018.

[7] M. R. Sethuraj and N. M. Mathew, *Natural rubber : biology, cultivation, and technology*, Elsevier, 1992.

[8] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[9] R. Socher et al., "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, pp.248–255, 2009.

[10] H. Caesar, J. R. R. Uijlings, and V. Ferrari, "COCO-Stuff: Thing and Stuff Classes in Context," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, pp.1209-1218, 2018.

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, pp.4510–4520, 2018.

[12] R. Fernández, H. Montes, J. Surdilovic, D. Surdilovic, P. Gonzalez-De-Santos and M. Armada,"Automatic detection of field-grown cucumbers for robotic harvesting," *IEEE Access*, vol. 6, pp.35512–35526, 2018.

[13] A. Silwal, J. R. Davidson, M. Karkee, C. Mo, Q. Zhang and K. Lewis, "Design, integration, and field evaluation of a robotic apple harvester," *Journal of Field Robotics*, vol. 34, no. 6, pp. 1140–1159, 2017.

[14] A. Leu et al., "Robotic green asparagus selective harvesting," in *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2401–2410, Dec.2017.

[15] L. Shao, X. Chen, B. Milne, and P. Guo, "A novel tree trunk recognition approach for forestry harvesting robot," *2014 9th IEEE Conference on Industrial Electronics and Applications*, Hangzhou,, pp.862–866, 2014.

[16] J. Lv, D. A. Zhao, W. Ji, Y. Chen, and H. Shen, "Design and research on vision system of apple harvesting robot," *2011 Third International Conference on Intelligent Human-Machine Systems and Cybernetics*, Zhejiang, vol. 1, pp.177–180, 2011.

[17] Y. Zhao, L. Gong, Y. Huang, and C. Liu, "A review of key techniques of vision-based control for harvesting robot," *Computers and Electronics in Agriculture*, vol. 127, pp. 311–323, 2016.

[18] C. Lehnert, I. Sa, C. McCool, B. Upcroft, and T. Perez, "Sweet pepper pose detection and grasping for automated crop harvesting," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, pp.2428–2434, 2016.

[19] K. Kusumam, T. Krajník, S. Pearson, T. Duckett, and G. Cielniak, "3D-vision based detection, localization, and sizing of broccoli heads in the field," *Journal of Field Robotics*, vol. 34, no. 8, pp.1505–1518, 2017.

[20] K. R. Vijayakumar et al., "Revised international notation for latex harvest technology," *Journal of Rubber Research*, vol.12, no.2, pp.103–115, 2009.

[21] S. J. Soumya, R. S. Vishnu, R. N. Arjun, and R. R. Bhavani, "Design and testing of a semi automatic rubber tree tapping machine (SART)," *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, Agra, pp.1–4, 2016.

[22] T. Veena and R. Kadadevaramath, "DESIGN AND FABRICATION OF AUTOMATIC RUBBER TAPPING MACHINE," *IOP Conf.*, 2016.

[23] J. V. C. Maliackal, K. A. Asif, P. A. Sajith, and S. K. Joseph, "Advanced Rubber Tree Tapping Machine," vol. 2, no. 5, pp. 253–263, 2017.

[24] S. Simon, "Autonomous navigation in rubber plantations," *2010 Second International Confer-*

ence on Machine Learning and Computing, Bangalore, pp.309–312, 2010.

[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.38, no.1, pp.142–158, Jan. 2016.

[26] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, pp.1440–1448, 2015.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.779–788, 2016.

[28] W. Liu et al., "SSD: Single shot multibox detector," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.

[29] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark Analysis of Representative Deep Neural Network Architectures," in *IEEE Access*, vol.6, pp.64270-64277, 2018.

[30] J. Huang et al., "Speed/accuracy trade-offs for modern convolutional object detectors," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, pp.3296-3297, 2017.

Rattachai Wongtanawijit is a PhD student at Computer Engineering at Prince of Songkla University (PSU). He received his honored bachelor's degree in Computer Engineering at PSU in 2015.

Thanate Khaorapapong is currently a professor and an associate department head for research and Graduate Studies at Computer Engineering, PSU. He received his doctoral degree in Doctorat Systems Automatiques Automatism at INPT/ ENSEEIHT France in 2001. His research interests includes image processing, robotics, and control system.