2950 Niles Road, St. Joseph, MI 49085-9659, USA
269.429.0300 fax 269.429.3852 hq@asabe.org www.asabe.org

An ASABE Meeting Presentation

DOI: https://doi.org/10.13031/aim.201901125

Paper Number: 1901125

Leveraging Deep Learning and RGB-D Cameras for Cooperative Apple-Picking Robot Arms

Hemanth Sarabu [1], Konrad Ahlin [2], Ai-Ping Hu [3]

[1] School of Computational Science and Engineering,

Georgia Institute of Technology, Atlanta, GA, USA

hemanth.sarabu@gtri.gatech.edu

[2] Georgia Tech Research Institute, Atlanta, GA, USA

konrad.ahlin@gtri.gatech.edu

[3] Georgia Tech Research Institute, Atlanta, GA, USA

ai-ping.hu@gtri.gatech.edu

Written for presentation at the

2019 ASABE Annual International Meeting
Sponsored by ASABE
Boston, Massachusetts
July 7–10, 2019

ABSTRACT.

We report on a novel method of using two cooperating robot arms to pick apples in a typical unstructured orchard. Each robot arm is equipped with an Intel Realsense D435 RGB-D (color plus depth) camera at its wrist. The first robot arm (termed the "Search arm") is used to detect apples and also to survey a given orchard tree's volumetric space for clear paths. The second, "Grasp arm", is positioned relatively closer to the tree and is designated with approaching fruit with the intent to harvest. A custom-trained deep learning-based object detection algorithm called YOLO (You Only Look Once) is used for finding apples in the cluttered scene. Apple location and clear path information is encoded into a graph and used for planning. The arms are driven by a finite state machine with information provided by the graph. Computer simulation and real world experimental results are reported. Based on our results to date, solutions are proposed to improve robustness, apple localization, and to minimize average time to pick all feasible apples.

Keywords. Apple Harvest, Agricultural Robot, Autonomous, Cooperative, Deep Learning, Image Processing, Manipulator, Path Planning, Unstructured.

Introduction

We report progress made on prior work by Ahlin, K. J., Hu, A. P., & Sadegh, N. (2017) towards the development of collaborative apple-picking robots. The Void Space Algorithm inspired by visual-servoing was proposed and tested in simulation using a Kinova robot arm with a monoscopic camera attached in an eye-in-hand configuration. In this paper, we propose improvements to the previously used techniques by adopting a deep-learning based apple detection algorithm and depth-sensing capabilities for apple localization and path planning. The idea of Void Space Planning is extended, improving on shortcomings associated with being memoryless and reactive, by incorporating a graph structure and finite state machine into the planning framework.

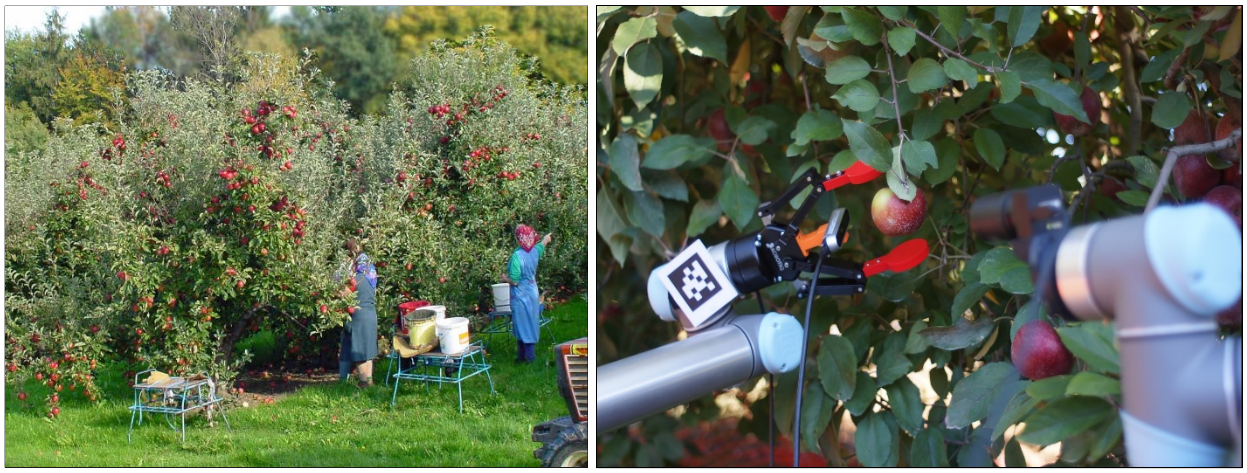


Figure 1: Manual apple picking (Left), Testbed apple picking (Right)

System Overview

The robotic system contains the following hardware elements:

- Grasp arm: 6-DOF UR5 arm with eye-in-hand RGB-D camera positioned in close proximity to the target apple tree with the intention of reaching for and harvesting apples
- Search arm: 6-DOF UR5 arm with eye-in-hand RGB-D camera positioned away from target tree and designated with tasks that aid the Grasp arm discover apples and navigate to them
- RGB-D Cameras: Intel Realsense D435
- Computer: Nvidia Jetson TX2

The logic architecture is implemented using ROS. Primary components include a perception module, planning module and a state machine. The perception module is responsible for the detection and localization of apples in addition to aiding the planning module search for paths. The planning module consists of the graph structure and a collection of algorithms to construct a lower-dimensional representation of the environment and for planning high-level tasks. The state machine is used to realize desired sequential behavior from the robotic system.

Graph Structure

The Void Space planning paradigm is augmented by the incorporation of a graph structure for memory and planning. It affords the system the ability to build and maintain a low-dimensional representation of its surroundings. The graph G is parameterized by nodes and edges (N, E) in the global coordinate frame:

1. Nodes: Nodes are used to represent two types of entities; physical location of an apple or a potential waypoint within the workspace of the Grasp arm. Naturally, every node object is imbued with a *location* attribute and a *type* attribute. Apple nodes are updated with an *estimated size* attribute. This is used for target localization when depth sensor readings are unreliable. Apple nodes can either be blue or red; a red node is used to indicate that the Grasp arm has registered the associated apple and deemed it to be within its workspace, thus establishing a candidate path to it. A blue node is used to indicate that the associated apple has been found in the workspace of the Grasp arm by the Search arm and not the Grasp arm. Hence, blue nodes are spawned as lone nodes with no edges to the remainder of the graph. Non-apple nodes, indicating used or candidate waypoints for the Grasp arm, are shown in green.
2. Edges: These are used to connect nodes with each other. Each edge is weighted by the Euclidean distance between the nodes it connects. Edges can either be green or red to designate direct connectivity or visibility respectively: green edges (connectivity) are used to represent candidate paths or traversed paths for the Grasp arm and connect green nodes. Similar to the former type, red edges (visibility) are used to record which apples (red nodes) are visible from a given position (green node). Hence, red edges only connect red nodes to green nodes.

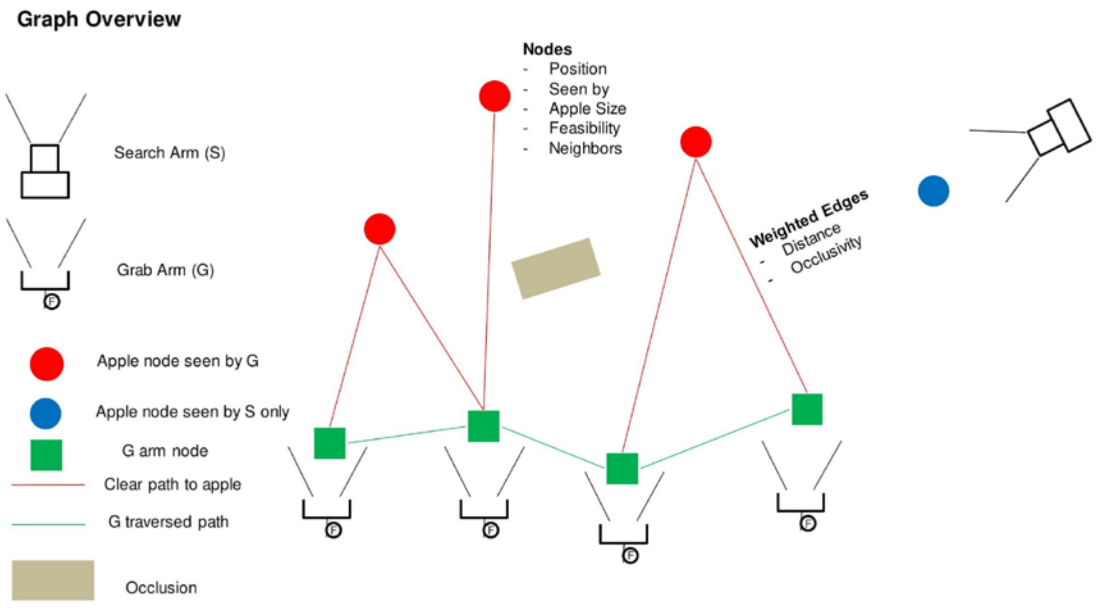


Figure 2: Overview of graph structure

Both arms play an active role in populating and updating the graph with newly discovered nodes and edges. The Grasp arm spawns a green node every time it moves to a new location. A red node is spawned when it detects a previously undiscovered apple that lies within its workspace. It also draws a red edge from itself to the node. The Search arm spawns a blue node in the graph if it detects a previously undiscovered apple that is within the Grasp arm's workspace. If the Grasp arm verifies that no apple is detected at a location of a blue node (poor detection/localization or if the apple has already been picked), it changes the node from red to green. This is done to reuse the node as a waypoint in future planning. The Grasp arm is limited to traverse within its connected subgraph and any nodes outside this region are ignored until a connection is made by either arm. In contrast, the Search arm is free to traverse outside the subgraph to connect *lone* blue nodes to G's configuration space whilst employing specialized scanning maneuvers to maximize apple discovery.

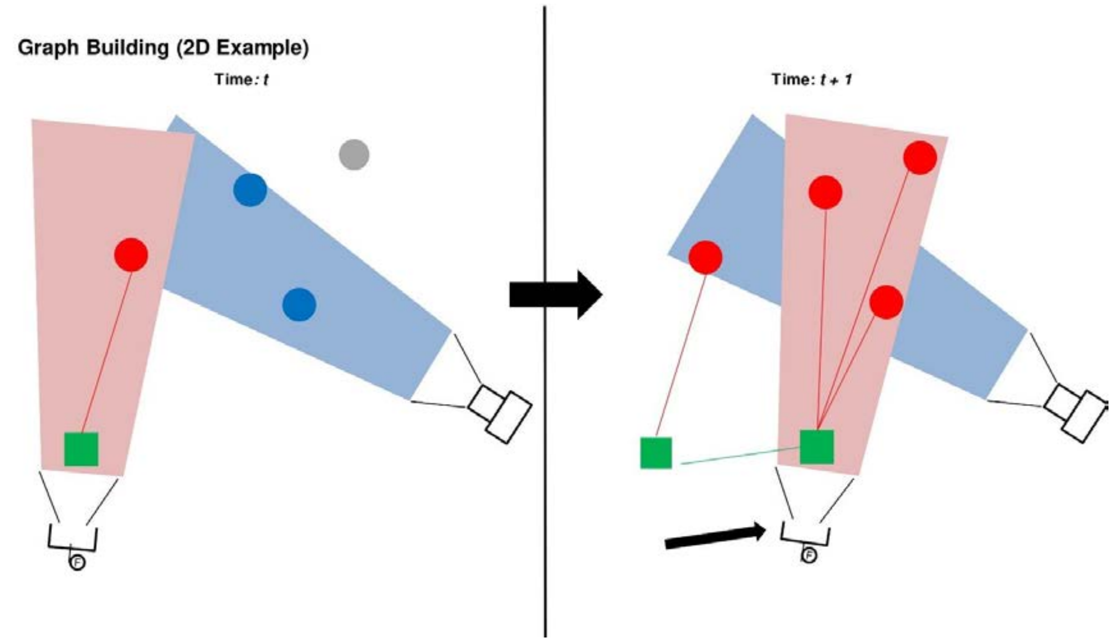


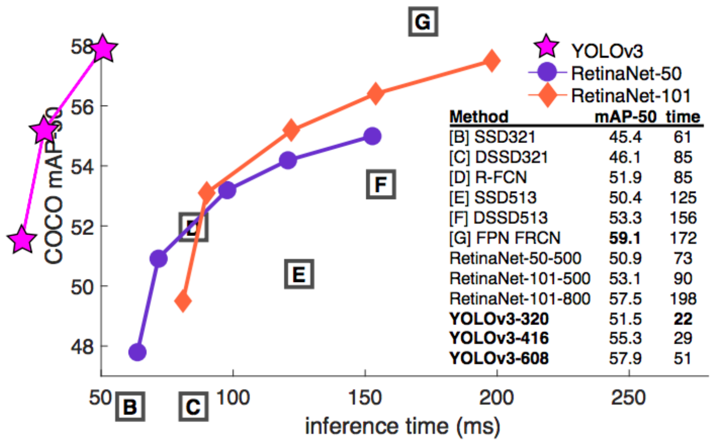
Figure 3: Example of graph being populated and updated

Apple Detection and Localization

In our prior work (Ahlin et al, 2017), cameras used on both arms were monoscopic (RGB only). The planning architecture employed a mix of classical image-processing techniques such as color-based filtering, contour generation and Hough transforms to detect and localize apples in pixel space. This information was then processed to estimate the positions of different apples in three dimensional space using a hard-coded nominal apple diameter and the pinhole camera model. However, it was seen that the detection algorithm was sensitive to lighting conditions (time of the day and weather) and required frequent parameter tuning. A literature review was conducted to survey how the state-of-the-art fruit detection algorithms are able to overcome these hurdles (Ji et al., 2012; Kapach et al., 2012; Wei et al., 2014; Bargoti et al., 2016; Liu et al., 2016; Sa et al., 2016; Stein et al., 2016). It was found that while a large majority of the work focuses on using classical techniques to extract handcrafted features from images, CNN-based architectures, that have been trained to extract salient features automatically, are picking up popularity rapidly. The detection algorithm adopted for this project is called YOLO (You Only Look Once): Real-Time Object Detection.

How YOLO works

According to the original paper on the algorithm by Redmond et al. (2016), most prior architectures for object detection re-purpose classifiers to perform detection across a range of spatial scales; a sliding window is used to convolve through the image with the intention of object detection. This procedure is repeated for different window sizes, naturally rendering the algorithm computationally expensive. The approach YOLO takes is different; object detection is framed as a regression problem where a single neural network is use to predict both bounding boxes and class probabilities of the objects detected in them. As a neural network is utilized to derive all the bounding boxes and class probabilities in an image in a single evaluation, the paper suggests that the architecture can be optimized end-to-end for detection performance. For applications needing real-time object detection, YOLO is an attractive solution with a significantly higher frame processing rate. See Figure 4 for comparison of mean Average Precision and Inference Time between YOLOv3 and prior work.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 4: YOLOv3 vs state-of-the-art (Redmon et al., 2016)

Motivation to adopt YOLO for the project are as following:
1. Reasonable accuracy for computational requirement
2. Suitable for real-time use without the need for desktop GPUs (can be run on mobile units such as the Nvidia Jetson TX2)
3. Widely used architecture with well documented results and optimizations
4. Multiple open-source implementations available
5. Adding or training on new datasets is relatively simple

Integrating YOLO

A ROS-wrapped version of YOLO, written using a C-based deep learning framework called Darknet, is currently integrated into the object detection module. Two Darknet nodes are running at any given time of operation, one for each arm (or camera). The Darknet nodes subscribe to their respective arm's raw RGB image topics and produce a list of objects detected in the images, their bounding boxes and their classification probabilities. The Darknet nodes are able to infer at a frame rate of $\approx$ 5 FPS during operation.

The Darknet implementation comes with many sets of pre-trained weights. The COCO dataset (Lin et al., 2014) weights were used as they were already trained to detect 80 common objects amongst which apples were one. These pre-trained weights yielded inconsistent performance in-situ testing; depending on the time of day, the type of apples used (real or fake), the algorithm would fluctuate between being able to detect most apples in the scene to none. Shown in Figure 5 are the prediction results on images taken on real apples in the lab (indoors). The two images shown are quite similar, yet the detected apples in both images are different. This is an indication of the pre-trained model's sensitivity to minor changes in the scene and occlusions.

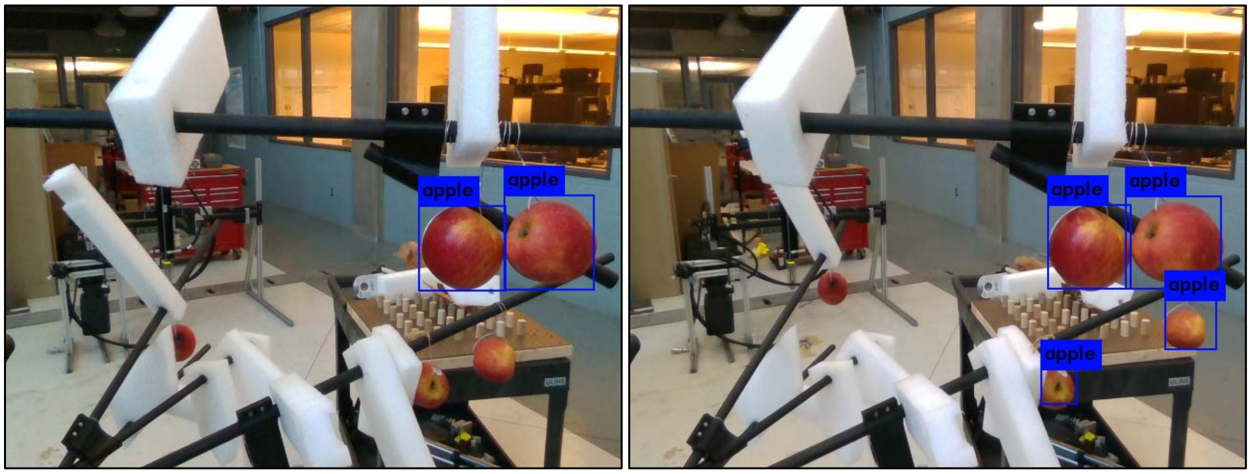


Figure 5: Inconsistent performance between similar frames

Shown in Figure 6 are prediction results of images that were taken in an apple orchard around two months prior to harvest season. The apples are visibly different compared to those used in experiments conducted in the lab (See Figure 5). It is seen that the weights fail to detect all except one apple in the first image and all of them in the second. This was the general behavior in performance that the pre-trained weights displayed on images taken during that trip. It is most likely because the apple occurrences that the COCO dataset contains are biased to certain types of apples that the weights are untrained to detect different variants of the apple object type it has learned. A contributing factor to the drop in detection performance in these images could be the loss of contrast in the frame due to the harsh lighting from an overhead sun.

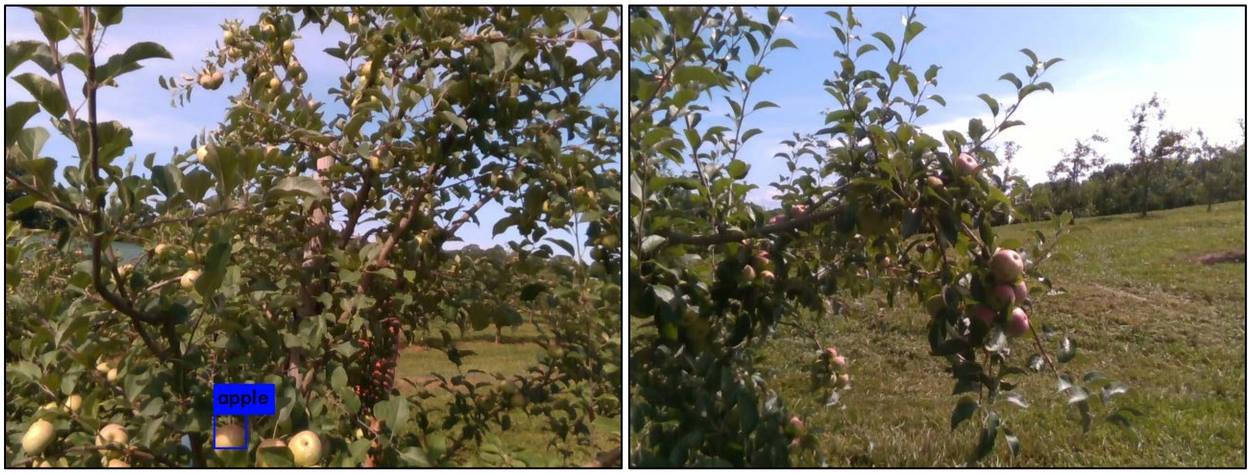


Figure 6: Poor detection performance outdoors and on different than expected apple types

To summarize, identified situations where detection is lacking:

1. Scenes where objects are back lit (low saturation and contrast)
2. Apples are of a different type than expected
3. Apples are of a different color than expected
4. Apples are occluded (partially or fully)

In order to robustify the model for the situations outlined above, new datasets were generated indoors and outdoors using fake and real apples. Images were captured using an Intel Realsense D435 and a cellphone camera:

1. Fake apples in lab at different lighting conditions

2. Real (Fuji and orchard) apples indoor at different lighting conditions
3. Fake apples outdoor at different lighting conditions
4. Real (orchard) apples outdoor with direct sunlight and harsh lighting

The classification layers were modified such that only one label needs to be predicted (apple). Weights for training were initialized using COCO weights. All images were labeled manually using a GUI that allows for bounding boxes to be drawn on an image and labeled (Figure 7). 425 images were selected and randomly split into training (365) and held-out sets (60). The training images were used by the network to update weights and the held-out data was used to compute mAP (mean Average Precision) and IoU (Intersection over Union of bounding box).

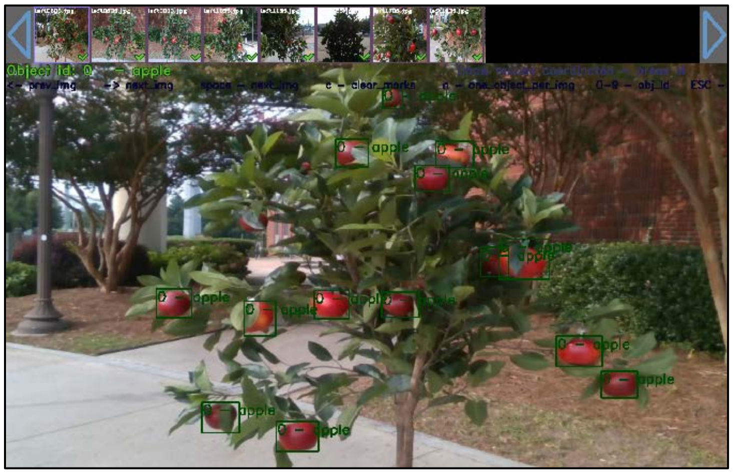


Figure 7: Using yolomark to add objects and draw bounding boxes

The updated model computed an average IoU (Intersection over Union of bounding box) of 0.86 on the held-out data. The model has been tested real-time conducting experiments in the high-bay and outdoors and is seen to perform well and issues encountered previously with pre-trained weights have not manifested. Experiments revealed that YOLO is able to detect over 92% of apples visible in a scene and produces bounding boxes of reasonable accuracy. This improvement in detection is sufficient for the purposes of this project.

Leveraging Depth-Sensing and Apple Localization

There are primarily two advantages to incorporating depth-sensing to the existing perception system. These are discussed in this section. The depth-sensing capability is brought by two Intel Realsense D435 (Figure 8) cameras mounted on the Grasp and Search arms (replacing the previously used Point Grey RGB cameras). The data consumed from the D435s are aligned RGB and depth images.

Figure 8: Intel Realsense D435

In our previous work (Ahlin et al, 2017), apples were detected in pixel space and localized in three dimensional space using the pin-hole camera projection model. This involved hard-coding a nominal apple radius to estimate depth to apple in the camera frame. This is a fair assumption as long as the variance in estimated apple size sizes is small and the nominal apple size is known. Experiments in the lab revealed that this is not the case in practice and a more robust localization method is required. This is where a reasonably accurate depth measurement can be of tremendous assistance. Similar to the previous pipeline, apples can be detected in RGB space. Depth measurements in the apples' bounding boxes can be used to more accurately determine their position when fused with their location in pixel space.

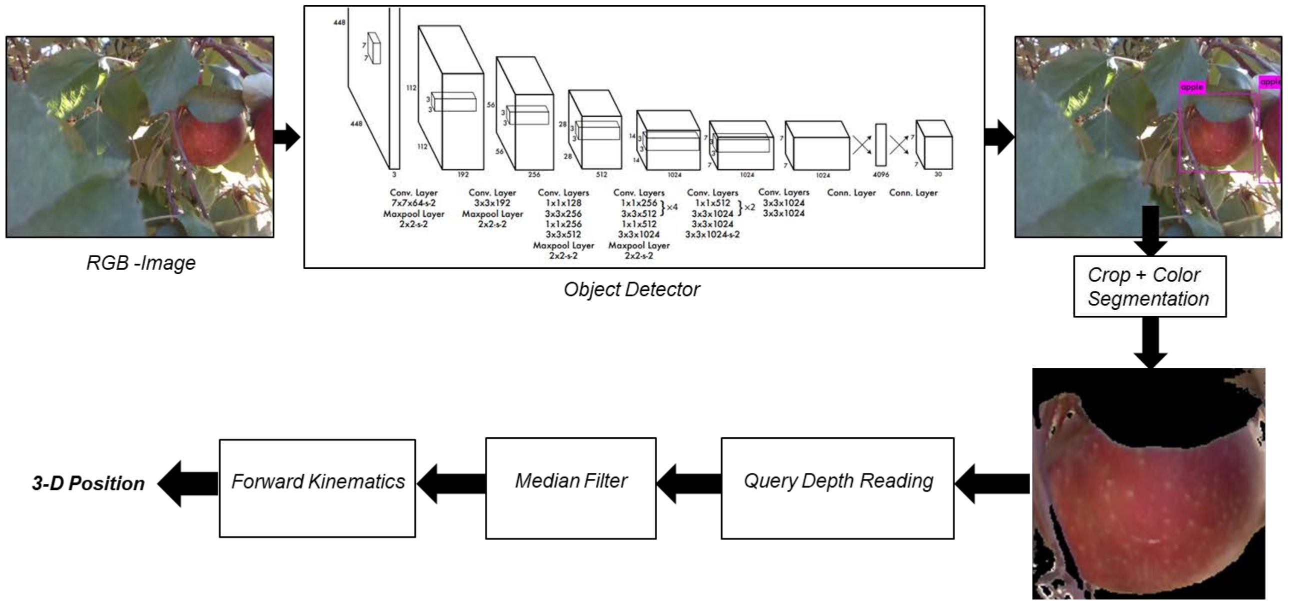

RGB -Image
Object Detector
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer
apple
Crop + Color Segmentation
Query Depth Reading
Median Filter
Forward Kinematics
3-D Position

Figure 9: Apple Detection and Localization Pipeline

The apple detection and localization pipeline is illustrated in Figure 9. The RGB image is fed into the object detector (YOLO Darknet ROS node) that produces a list of bounding boxes and probabilities for apples found in the image. These bounding boxes are used to take a cropped section of the RGB image containing an apple and filter the section based on color. This is done to create a mask than can then be applied to the same section of the aligned depth image and retrieve a collection of depth values associated with the apple. The median filtered value from the obtained depth values is used with the forward kinematics of the camera's respective arm to estimate the apple's 3-D position. More sophisticated methods for localization can be used but the technique outlined above has proven to be sufficient for the purposes of this project. It is important to note that this technique fails when the depth camera is unable to pick up reliable depth readings at the section of the image associated with an apple. In these cases, we revert to using the pin-hole camera projection model to estimate the depth to the apple based on a nominal diameter.

Results

We showcase results from simulation and experimental testing. In the images shown in this section, the Grasp and Search arms are labelled G and S respectively.

Simulation Results

The simulation environment is constructed in VREP and features two 6-DOF UR5 robot arms located 1.2 meters apart. Each arm's end-effector is fitted with a simulated eye-in-hand RGB-D vision sensor. The sensor is configured to mimic the intrinsic properties of the Realsense D435 cameras used in experiments. Red spheres are used in place of apples to populate a three dimensional CAD model of a tree. The spheres are segmented in the RGB image leveraging color-based filtering, contour detection and Hough transforms. Using this information and known arm poses, depth images simulated by the sensor are processed to obtain the spheres' locations in the global coordinate frame.

Figure 10 portrays an example case with two apples (*A* and *B*) positioned in the workspace of the Grasp arm such that during its scan routine, one (A) is easily discovered while the other remains obscured (*B*). *B*, initially, is only visible to the Search arm. The simulation begins with both arms assuming an initial pose before the Grasp arm performs a series of motions to scan the environment for apples. The Grasp arm discovers sphere *A* and spawns a red node in the graph structure. This is visualized in Figure 11; RGB feed from the Search and Grasp arms are shown in the upper-left and lower-left panels respectively and the graph structure in the right. Nodes traversed by the Grasp arm during the simulation are colored green.

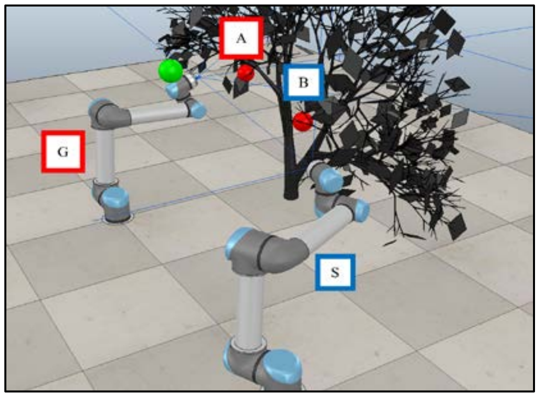


Figure 10: Simulated Scene in VREP

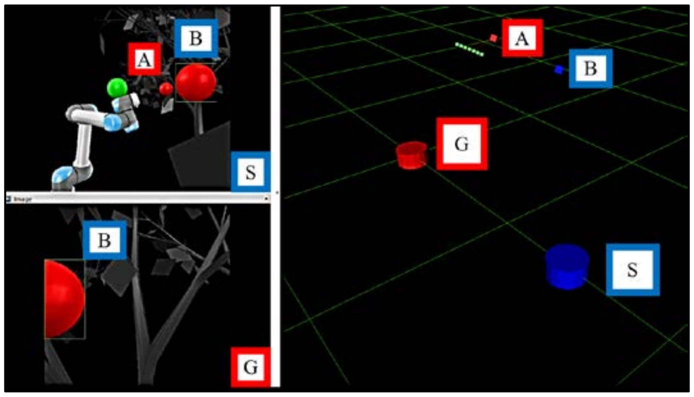


Figure 11: RGB Camera feeds (left) and graph (right)

Once the scan is complete, the Grasp arm uses visual servoing to mime an *approach*-and-*grasp* maneuver towards sphere *A*. The node corresponding to *A* is updated to reflect that no spheres occupy the space and is then colored green.

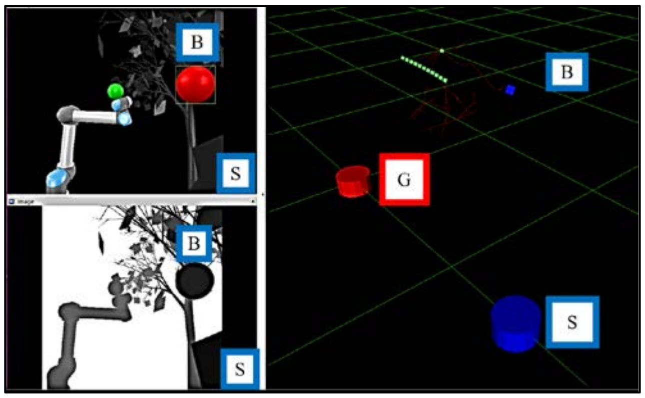


Figure 12: Node *A* picked and colored green

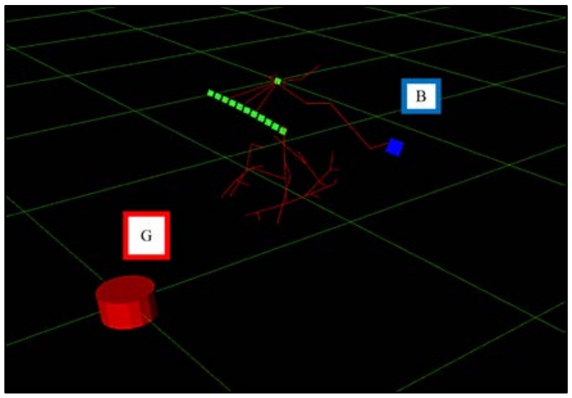


Figure 13: Viable path from Grasp arm to node *B*

Concurrently, S performs its own scan routine and discovers a previously unregistered sphere (node *B*). In Figure 12, the Search arm's RGB and depth feeds are shown in the left panels. Once G completes its grasp maneuver, S generates a viable path using the depth map between node *B* and the Grasp arm's current location. The Search arm employs Heuristic Positioning and In-Image RRT-Connect, algorithms tailored for this application by Sarabu, Ahlin and Hu (2019), to search for a feasible path by direct operation on depth images (without mapping). These techniques are based on the original RRT-Connect algorithm proposed by Kuffner and LaValle (2000). The depth image shown in Figure 12 is consumed by In-Image RRT-Connect to connect the lone blue node (*B*) to the rest of the graph (Figure 13). Once connected, the shortest path to *B* from the Grasp arm's location is found by calling Dijkstra's shortest path first algorithm.

Experimental Results

Similar to the simulated environment, the experimental setup consists of the following elements:

- Two UR5 6-DOF robot arms
- Intel Realsense D435 RGB-D cameras placed in the arm grippers (eye-in-hand configuration)
- Artificial apple trees and apples to represent an agricultural environment

The case presented in simulation is replicated experimentally (note change in placement of apples in Figure 14). Here node

A is positioned in front of the Grasp arm (*G*) such that it is detected during its predetermined scan motion and node *B* is placed such that it is only registered by the Search arm. Two instances of YOLO are run simultaneously to service each arm independently. This allows apples to be detected and registered by both arms.

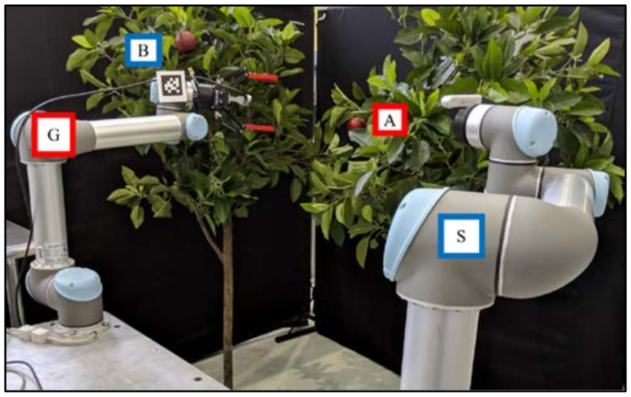


Figure 14: Experiment setup

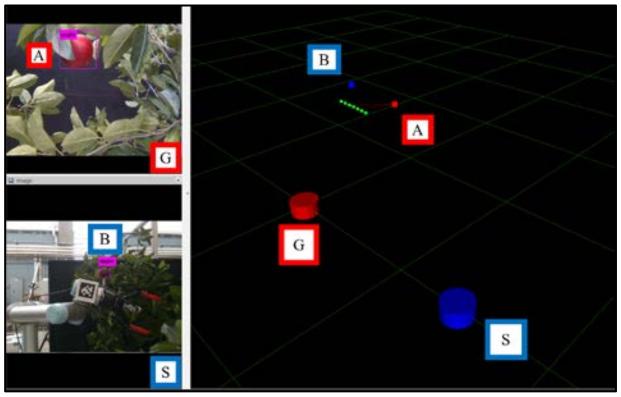


Figure 15: Apples detected using YOLO and added to graph

Figure 15 illustrates apple *A* being detected by the Grasp arm and added into the graph (red node) and apple *B* (blue node) by the Search arm. The upper and lower side panels display a snapshot of the RGB feed to the grasp and Search arms respectively. The graph being generated is shown to the right. Upon finishing its scan routine, the Grasp arm employs Dijkstra's algorithm to generate a path to node *A*. It completes the *approach*-and-*grasp* maneuver and updates node *A* to green. Shown in Figure 16 are snapshots of RGB and depth feeds (left) to the Search arm and the state of the graph after the Grasp arm has picked the apple *A* (right). The Search arm uses the depth image and In-Image RRT-Connect to add a series of feasible nodes such that *B* is connected to the rest of the graph (Figure 17). In-Image RRT-Connect relies on the idea of sampling for points in the depth image to form connections. Therefore, successful path generation in cases such as this is contingent on the availability of depth information to form connections.

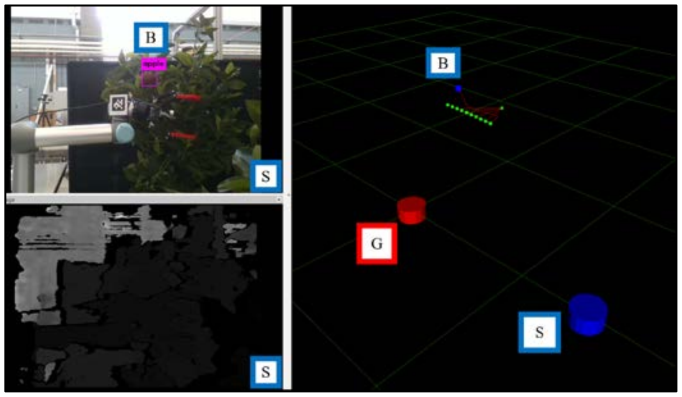


Figure 16: Node *A* picked and colored green

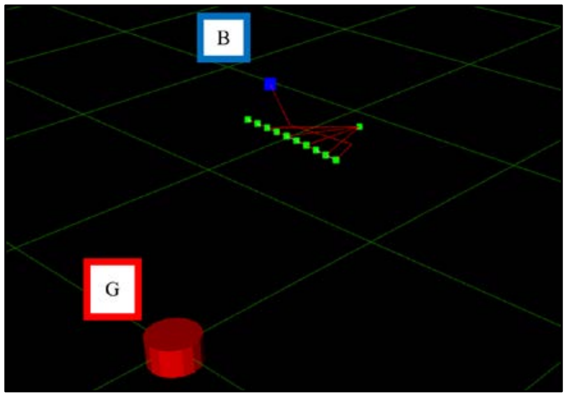


Figure 17: Viable path from Grasp arm to node *B*

Conclusion

The challenges surrounding the development of harvesting robots for unstructured agricultural environments remain widely unsolved to this day. In this work we propose and study perception and planning related techniques bolstered by advances in deep learning and depth sensing technologies to realize cooperative behavior between robotic arms in an artificial orchard. Results indicate that these technologies can be readily applied to add robustness to fruit discovery and maximize the percentage of fruit that can be harvested through advanced planning techniques.

References

Ahlin, K. J., Hu, A. P., & Sadegh, N. (2017). Apple picking using dual robot arms operating within an unknown tree. In 2017 ASABE Annual International Meeting (p. 1). American Society of Agricultural and Biological Engineers.

Alexey. Yolo-v3 and Yolo-v2 for Windows and Linux. Retrieved from https://github.com/AlexeyAB/Darknet

Alexey. (2017). Windows & Linux GUI for marking bounded boxes of objects in images for training Yolo v3 and v2. Retrieved from https://github.com/AlexeyAB/Yolo_mark

Bargoti, S., & Underwood, J. (2017). *Deep fruit detection in orchards.* Paper presented at the 2017 IEEE International Conference on Robotics and Automation (ICRA).

Ji, W., Zhao, D., Cheng, F., Xu, B., Zhang, Y., & Wang, J. (2012). Automatic recognition vision system guided for apple harvesting robot. *Computers & Electrical Engineering, 38*(5), 1186-1195.

Kapach, K., Barnea, E., Mairon, R., Edan, Y., & Ben-Shahar, O. (2012). Computer vision for fruit harvesting robots–state of the art and challenges ahead. *International Journal of Computational Vision and Robotics, 3*(1/2), 4-34.

Kuffner Jr, J. J., & LaValle, S. M. (2000, April). RRT-connect: An efficient approach to single-query path planning. In ICRA (Vol. 2).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., . . . Zitnick, C. L. (2014). *Microsoft coco: Common objects in context.* Paper presented at the European conference on computer vision.

Liu, X., Zhao, D., Jia, W., Ruan, C., Tang, S., & Shen, T. (2016). A method of segmenting apples at night based on color and position information. *Computers and Electronics in Agriculture, 122*, 118-123.

Redmon, J. (YOLO: Real-Time Object Detection). Retrieved from https://pjreddie.com/darknet/yolo/

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You only look once: Unified, real-time object detection.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Robotics, E. Z. L. YOLO ROS: Real-Time Object Detection for ROS. Retrieved from https://github.com/leggedrobotics/darknet_ros

Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., & McCool, C. (2016). Deepfruits: A fruit detection system using deep neural networks. *Sensors, 16*(8), 1222.

Sarabu, H., Ahlin, K. J., & Hu, A. P. (2019, July). Graph-Based Cooperative Robot Path Planning in Agricultural Environments. In 2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) (In-Press).

Stein, M., Bargoti, S., & Underwood, J. (2016). Image based mango fruit detection, localisation and yield estimation using multiple view geometry. *Sensors, 16*(11), 1915.

Wei, X., Jia, K., Lan, J., Li, Y., Zeng, Y., & Wang, C. (2014). Automatic method of fruit object extraction under complex agricultural background for vision system of fruit picking robot. *Optik-International Journal for Light and Electron Optics, 125*(19), 5684-5689.