Proceedings of the 2019 IEEE/ASME
International Conference on Advanced Intelligent Mechatronics
Hong Kong, China, July 8-12, 2019

Graph-Based Cooperative Robot Path Planning in Agricultural Environments

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

***Abstract*— This paper describes a method of using dual robot arms to cooperatively pick apples in an unstructured orchard environment. Each arm is equipped with an RGB-D (color-plus-depth) camera in an eye-in-hand configuration. The first robot arm, termed the *Grasp* arm, is positioned relatively close to the tree and designated with picking apples. It uses its camera to locate apples that are within view and also within reach. The second *Search* arm, is located nearby and is used to detect apples that are hidden from the grasp arm and to plan a clear path for it to those fruits using a method based on rapidly-exploring random trees. Fruit location and clear path information is encoded into a graph representation that is expressive enough to encode memory and lends itself to various decision-making algorithms. It embodies the idea of using clear paths for planning, as opposed to mapping obstacles, hence maintaining a lower-dimensional representation. Computer simulation and experimental results are presented based on a preliminary implementation.**

Fig. 1: Testbed dual robot arm system in an apple orchard.

I. INTRODUCTION

The approach taken to automated fruit harvesting usually divides the task into the following operations: fruit detection and localization followed by robot path planning and control [13]. A survey of popular sensor modalities, detection and control algorithms, as well as current challenges has been presented recently in [15] and [14]. A review of prior work reveals that assumptions of planar canopies (trellises) and lack of obstacles are popular among research groups. It is also apparent that limited work has been done so far in investigating the possible benefits of using cooperative robots to augment target detection and path generation for this application space.

In this paper, we report on the progress of on-going work [1] to develop and implement path planning and control strategies for dual robot arms (a *grasp* arm and a *search* arm) tasked with picking fruit in a natural, unstructured apple orchard environment (see Fig. 1). The grasp and search arms are used to detect, localize, and approach apples while generating paths in a collaborative fashion. We use a unified graph to represent viable target apples and clear spaces between points, thus reasoning over a lower-dimensional representation than a traditional 3D occupancy map. Each arm has an RGB-D (i.e., color-plus-depth) eye-in-hand camera, used for perception and motion control via visual servoing.

[1]Hemanth Sarabu is a graduate student in the School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA USA hemanth.sarabu@gtri.gatech.edu

[2] Konrad Ahlin is a Research Engineer at the Georgia Tech Research Institute, Atlanta, GA USA konrad.ahlin@gtri.gatech.edu

[3] Ai-Ping Hu is a Principal Research Engineer at the Georgia Tech Research Institute, Atlanta, GA USA ai-ping.hu@gtri.gatech.edu

Our work includes elements in common with prior reported automated fruit picking research. In 2011, [4] proposed an apple harvesting robot design with an eye-in-hand camera and a laser range sensor that was able to achieve a success rate of over 90% using open-loop visual servoing. The average cycle time per apple was 7.1 seconds. Later in 2013, [8] used a modular hierarchical planning framework where apples were detected and recorded into a target list and a collision map including a compressed 3D point cloud data list with annotations of targets and obstacles was maintained. Various probabilistic motion planners were tested and their solution times were reported. In 2017, [7] designed a sweet pepper harvesting robot that used an eye-in-hand RGB-D camera and a predetermined scanning motion to build a 3D model of the environment. The sweet peppers were segmented in the image using color information and localized using a fitted parametric model. Two state machines were run concurrently where one is continuously processing segmented sweet peppers to calculate candidate grasping and cutting trajectories while the other plans and executes actions on available targets. It appears that this reactive system is heavily reliant on the predetermined scan pattern and no adaptive scanning and mapping techniques are used.

Three relevant examples of prior work have been found where collaborative robots have been used in fruit harvesting. In 2004, [9] proposed and tested a masterslave robot motion control system developing a GOTO algorithm and a FOLLOW algorithm for field vehicles. In 2014, [16] described algorithms that plan the assignment of melons to be harvested for a given number of robotic arms and their capabilities while Cartesian manipulators mounted on a rectangular frame carry the arms laterally across the crop

978-1-7281-2492-6/19/$31.00 ©2019 IEEE

bed. The paper assumes that the locations of melons are known a priori and that no sensing is required to detect and localize the targets. The planning problem is reduced to a k-colorable sub-graph problem and they have shown that in some situations a local search algorithm could produce near-optimal assignments. In 2017, [3] designed and tested a robotic system that used one manipulator to pick apples and a second one to collect the picked apples near the point of detachment. By adopting pick-and-catch over a pick-and-place routine, they have reportedly achieved a cycle time reduction of 50%.

We use dual robot arms each equipped with an eye-in-hand RGB-D camera working cooperatively to populate a unified graph representing candidate detected apple locations and clear paths to them. Our goal is to operate within a natural unstructured orchard for which no a priori information is assumed known and to achieve grasping of all apples within reach of the designated grasp arm. The graph representation is expressive enough to encode memory and lends itself to various decision making algorithms. It embodies the idea of using clear paths for planning, as opposed to mapping obstacles, hence maintaining a lower-dimensional representation. We describe our graph representation, a functioning state machine to schedule robot arm activities, and initial results of implementing a path planning method based on a variant of rapidly-exploring random trees (RRTs) [6], the latter for which there is prior application in automated fruit harvesting. For example, in 2015, [10] compared sampling-based path planners for the application of grape vine pruning using robotic arms and found that RRT-based algorithms displayed best overall performance (success rate and computation time) in unstructured settings. In 2019, [2] developed an RRT-based planning technique for an automated litchi-picking manipulator. The researchers biased new samples added to the tree towards the target in order to accelerate path generation and applied genetic and smoothing algorithms to optimize the proposed path.

In our dual arm approach, the grasp arm is only able to generate paths to apples that it discovers during its predetermined scan routine or while it effects actions to approach mapped apples. The search arm is employed to rectify this shortcoming by finding apples that are within the grasp arm's configuration space and generating paths from them to the grasp arm's connected sub-graph. This problem is well-suited for bidirectional search algorithms such as RRT-Connect [5], where trees are grown from both the source and the goal configurations to generate paths.

II. METHOD

The proposed robotic system adopts a look-then-move scheme during most operational states and an open-loop visual servoing control scheme during the final reach-and-grasp sequence. Motion to targets is executed in finite steps, interleaved with perception related tasks.

A. System Overview

Grasp Arm: The grasp arm (G) is a 6-DOF robotic manipulator outfitted with an eye-in-hand RGB-D camera. This arm is designated to approach target apples and mime a picking a motion.
Search Arm: The search arm (S) is identical to G, except it is not equipped with a gripper. It is used to assist G in finding apples and generating viable paths.
Graph: An internal graph representation is constructed based on apples (nodes) and clear paths (edges) discovered by arms during operation. High-level planning decisions (such as which apple to pick at a given time) are made upon reasoning over this graph.
Finite State Machine: A finite state machine (SM) designed to realize the desired behavior is implemented using ROS Actionlib and SMACH.

B. Graph

The graph structure (Fig. 2) has the following attributes:
Nodes: A node in the graph can represent either an apple or a position that G's end-effector (camera and gripper) can move to. Every node in the graph possesses a location attribute that specifies its position in the global coordinate frame. Apple nodes that have been "seen" by G are colored red and those perceived only by S are colored blue. Green nodes denote positions in space that G has either been in or can move to. Apple nodes (blue and red) also carry a size attribute that is estimated and updated using simple machine vision techniques. Two types of edges are used to connect nodes with one another.
Connectivity: These are the first type of weighted edges; they are drawn between nodes that G has traversed through. An edge is weighted by the Euclidean distance between the nodes it is connecting. These edges are shown in green.
Visibility: These are the second type of weight edges; when G is able to "see" an apple, one of these edges is drawn between the nodes corresponding to G's current location and the apple that it sees. The weights are, again, set to Euclidean distances. These edges are shown in red.

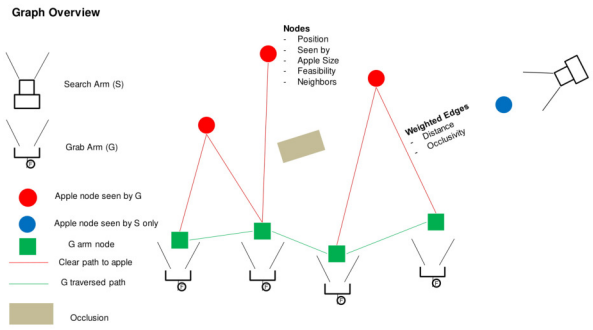

Fig. 2: Graph overview.

C. Role of Grasp Arm in Graph Building and Maintenance

Apart from grasping and dropping apples, G is responsible for the following actions:

- G spawns a new green node every time it moves to a novel position.
- When G sees an undiscovered apple within its configuration space, it spawns a red node in the graph and draws a red edge to it from its current node.
- When G sees an apple that already has a corresponding node in the graph, it draws a red edge from its current position to the apple node that it sees, therefore increasing the degree of the apple node. It also colors that node red, irrespective of its previous color.
- When G does not see an apple at a location where it expects to see one (due to either apple being picked already, poor localization, or poor detection), it changes the node's color from red to green. This is done to reuse that node for planning and sampling purposes in the future.
- G is limited to traverse within its connected sub-graph $\mathcal{C}$. Any nodes spawned outside of the connected component are ignored until a connection is made between them and the sub-graph.

D. Role of Search Arm in Graph Building and Maintenance

- When S sees an undiscovered apple within G's configuration space, it spawns a blue node in the graph.
- It conducts its own scanning maneuvers to maximize apple discovery.
- It is responsible for connecting "lone" blue nodes to the sub-graph that G is limited to traverse.

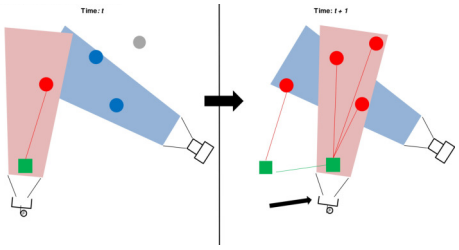

Fig. 3: Graph building schematic shown in 2D.

E. Desired Cooperative Behavior

The cooperative behavior desired of the dual robot arm system is explained in this section. If G's 6-DOF workspace is denoted by $\mathcal{G}$, then we aim to maximize the number of apples in $\mathcal{G}$ that are registered and harvested by G. This involves discovering these apples (spawning nodes either red or blue) and then turning them red by some means. Explicitly, G spawns red nodes when it sees an apple, indicating a clear path to the node. This (given our assumptions) guarantees that the apple can be picked. Blue nodes that have been spawned by S and not registered by G are not within the connected sub-graph $\mathcal{C}$, although they are in $\mathcal{G}$. In order for G to be able to pick apples corresponding to blue nodes, a viable path connecting them to $\mathcal{C}$ needs to be generated. If the system is designed such that only G can generate paths to blue nodes by "looking" for them, this distracts G from the activity of picking already-registered apples within $\mathcal{C}$

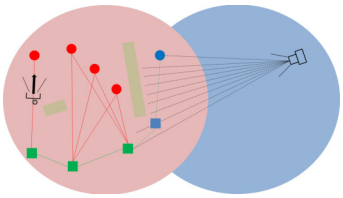

Fig. 4: S sampling for free space between blue node and green node.

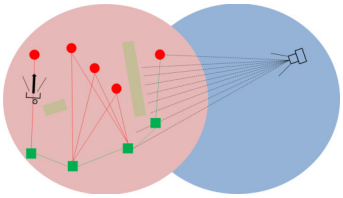

Fig. 5: S turning blue node red and connecting it to $\mathcal{C}$.

(existing red nodes). Instead, we propose using S to connect blue nodes that it has discovered to $\mathcal{C}$ using a sampling-based path planner.

To illustrate, shown in Figs. 4 and 5 is an example case where S complements G by turning a blue node red. The large red and blue circles represent the workspace of G and S, respectively. No assumptions of workspace overlap are made. The blue node is an apple that was detected by S and not confirmed by G, hence there exists no path connecting it to $\mathcal{C}$. S is able to use its depth sensing capabilities to sample for unobstructed free space between the blue node to the red nodes in $\mathcal{C}$. In this case, it has found a viable path by introducing an additional node (shown as a blue square) and using the proposed edges (shown as dashed green lines). The additional node is required to direct G to go around the rectangular obstacle shown in the figures. The proposed edges and node can be turned solid green and red, respectively, as the blue apple node is now connected to $\mathcal{C}$. We achieve this behavior by solving two sub-problems. The first entails positioning and orienting S to maximize the number of nodes that share their manifold with the blue node in the depth map. The second involves using the depth map in a sampling-based planner.

We take a heuristic approach to positioning S such that, should a path exist between the target blue node and the graph, our chances of finding one are maximized. When S is asked to generate a path from the blue node to the graph, it assumes a pose such that there is a clear path from itself to the blue node. A fixed number of samples are drawn from S's workspace by perturbing its end-effector's current pose and for each sample, the number of nodes in the graph that can be projected onto the image is evaluated. This provides an upper bound for the number of nodes that would be visible from the sampled pose in the absence of any occlusions. We accept the sample with the largest upper bound and have S assume the proposed pose. We then check, using the depth image, if a clear path is detected to the blue node. Using this idea we also count the number of clear paths detected to

other nodes in the graph and check if the number is larger than some threshold (m). This is a tune-able parameter. If the checks stated fail, we use the sampled pose with the next largest upper bound and repeat the process. The upper bound here is just a heuristic (indicator of promise) that we use in place of physically assuming each sampled pose for S and counting the number of nodes visible. Once S assumes a pose that satisfies the checks stated, RRT-Connect is employed to grow bi-directional trees and search for a path. If a path is not found within a fixed time allocated for a pose, S's pose is perturbed again and the previous steps are repeated to continue growing the same search trees dynamically from different vantage points. To this end, we present the Heuristic Positioning and In-Image RRT-Connect algorithm (HPIRC).

F. Proposed Algorithm to Connect Blue Nodes

$\mathcal{C}$: Connected sub-graph consisting of nodes (vertices) and edges
$\mathcal{S}$: $\mathcal{S} \subset \mathbb{R}^6$ workspace of S
$\mathcal{G}$: $\mathcal{G} \subset \mathbb{R}^6$ workspace of G
$\mathcal{H}$: $\mathcal{H} \subset \mathbb{R}^3$ Cartesian space bounds for RRT samples
q: $q \in \mathcal{H}$ Cartesian space sample for RRT
t: Unconnected target blue node
s: Search arm current pose
p: Sampled search arm pose
m: Threshold for minimum number of visible nodes for a search arm pose

Algorithm 1 Heuristic Positioning + In-Image RRT-Connect

```
1:  procedure CONNECT_GRAPHS(C, t)
2:      for j = 1 to J do
3:          POSITION_SEARCH_ARM()
4:          for k = 1 to K do
5:              τ_a.INIT(C), τ_b.INIT(t)
6:              q_rand ← RANDOM_CONFIG()
7:              if not(CONNECT(τ_a, q_rand) = Trapped) then
8:                  if (CONNECT(τ_b, q_new) = Reached) then
9:                      C ← τ_a ∪ τ_b
10:                     return C
11:             SWAP(τ_a, τ_b)
12:     return Failure
```

The functions EXTEND and CONNECT (**Algorithm 2**) are elaborated upon in the original paper on RRT-Connect [5] with descriptions on NEAREST_NEIGHBOR, NEW_CONFIG and RANDOM_CONFIG (**Algorithm 5**). The CHECK_VISIBLE function is used to check if a given sample q can be projected onto S (camera) at the current state and determine using the depth image if there exists a clear line of sight between itself and S. Similarly, using camera extrinsic and intrinsic properties, the CHECK_PROJECTION function checks, for a given pose of S and sample q, whether q's projection would fall in the S's image in the absence of obstacles. This information is used to calculate the aforementioned upper bound on the number of visible nodes for a given pose of S. The RANDOM_CONFIG (**Algorithm 5**) function is used to generate samples in the Cartesian workspace of G. To do so, it draws a sample from a uniform distribution in volume $\mathcal{H}$. Although we have used a simple sampling technique for this application, $\mathcal{H}$ itself can be a more expressive heuristic function that biases samples being drawn for the RRT planner. We have tuned $\mathcal{H}$ such that samples are not drawn too far from the trees (e.g behind the arms). Smart sampling heuristics can reduce the number of wasted samples and, therefore, speed up planning. If the sample lies in $\mathcal{G}$ and if it is visible to S (using depth), it is returned (rejected and re-sampled otherwise). The NEW_CONFIG function is used to generate a sample in the direction of q_{new} from q_{near} at some pre-detemined distance ϵ and add an edge between q_{new} and

Algorithm 2 Connect & Extend from RRT-Connect

```
1:  procedure CONNECT(τ, q)
2:      while True do
3:          S ← EXTEND(τ, q)
4:          if S ≠ Advanced then
5:              return S

6:  procedure EXTEND(τ, q)
7:      q_near ← NEAREST_NEIGHBOR(q, τ)
8:      if NEW_CONFIG(q, q_near, q_new) then
9:          τ.ADD_NODE(q_new)
10:         τ.ADD_EDGE(q_near, q_new)
11:         if q_new = q then
12:             return Reached
13:         else
14:             return Advanced
15:     return Trapped
```

Algorithm 3 Position Search Arm

```
1:  procedure POSITION_SEARCH_ARM
2:      {p_j}_{0:N} ∼ SAMPLE_SEARCH(s)
3:      {p_j}_{0:N} ← SORT_SAMPLES({p_j}_{0:N})
4:      for each p_j do
5:          MOVE_TO(p_j)
6:          if CHECK_VISIBLE(C, t) then
7:              break
```

Algorithm 4 Sampling Search Space

```
1:  procedure SAMPLE_SEARCH
2:      while True do
3:          p ∼ PERTURB(s)
4:          if p_i ∈ S and CHECK_PROJECTION(t, p_j) then
5:              break
6:          return p_i
```

Algorithm 5 Random Config

```
procedure RANDOM_CONFIG
    while True do
        q ~ uniform(H)
        if q ∈ G and CHECK_VISIBLE(q) then
            break
        return q
```

q_{near}. The accepted sample has to pass collision checks and must be in $\mathcal{G}$. We apply the collision constraint using the function CHECK_VISIBLE. The SAMPLE_SEARCH function (**Algorithm 4**) is used to draw samples obtained by random perturbation of S's camera pose that lie in $\mathcal{S}$ and satisfy CHECK_PROJECTION.

G. Finite State Machine

The states and transitions of the designed finite state machine (Fig. 6) are described briefly in this section.

INITIALIZE: This is the first state that is entered when the state machine (SM) is run. It initializes the arms, hardware handling objects and the graph.

SIMPLESCAN: This is a scanning state where the arms go through a predetermined scan pattern to start populating the graph with nodes and edges.

CONTINUE: In this state, the number of apples remaining in the graph is checked. If the number is larger than 0, the machine is transitioned into *PLAN*. Otherwise, the experiment is concluded as there are no more apple nodes remaining in the graph.

PLAN: The *PLAN* state runs Dijkstra's algorithm to compute shortest path to the nearest apple with the current G location as source (see 7). Unsurprisingly, the search is conducted within $\mathcal{C}$ as this is the space G is limited for traversal. If all apples within within $\mathcal{C}$ are exhausted, the machine is transitioned into the *EXTEND* state.

TOAPPLE: Once a target apple is provided, in this state, the grab arm is moved along the nodes in the path generated by the *PLAN* state. This is done until the parent node of the target apple is reached.

REACHGRAB: In this state, G is first made to look toward the target apple node for confirmation - it runs a detection and localization loop to confirm if an apple exists at the expected location. If an apple is detected at the location, G uses visual servoing to approach the targer. If this is successful it transitions into the *DROP* state. If no apple is detected, the corresponding node is changed from red to green and the machine is transitioned into the *CONTINUE* state.

DROP: The *DROP* state is used to perform a series of action to represent the grasping and dropping of the apple. As actual grasping of apples is outside the scope of this research, motions that mimic grasping are effected in this state. This includes G dropping the apple at a set home location. This can be modified to drop the apple upon disconnection.

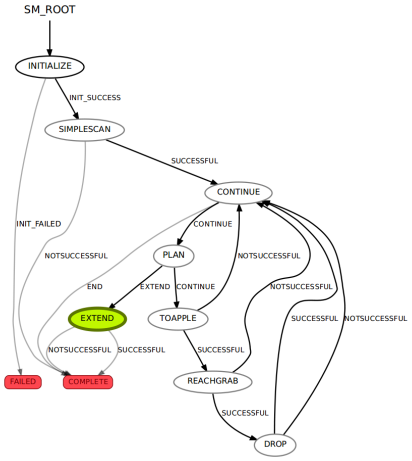

Fig. 6: Finite state machine.

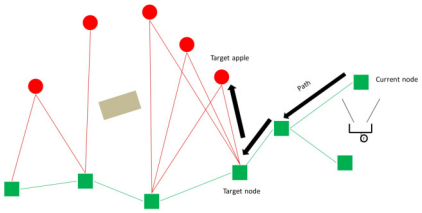

Fig. 7: Graph search using Dijkstra's algorithm.

EXTEND: The *EXTEND* state is used to run HPIRC (**Algorithm 1**) and generate paths to unconnected (blue) nodes.

While the SM is designed primarily to schedule tasks for G, the HPIRC algorithm is run concurrently on the search arm to continuously seek blue nodes and search for paths.

III. RESULTS

This section presents preliminary results from computer simulation and experimental testing.

A. Simulation

The simulation set up in VREP includes two 6-DOF robot arms (UR5 models) with bases positioned approximately 1.2 metres apart and each arm equipped with an RGB-D camera. We use a 3D model of a tree populated with red spheres (as proxies for apples) to test and develop the robotic system in a simulated unstructured setting. Simple machine vision techniques involving color-based image segmentation and Hough transforms from OpenCV are applied to RGB images to detect red spheres. They are localized in a global coordinate frame using depth information and known camera

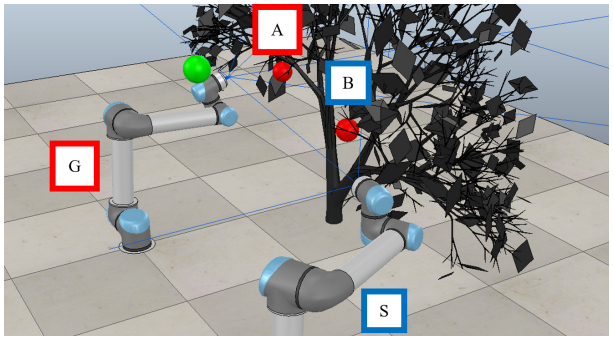

Fig. 8: Simulation setup in VREP.

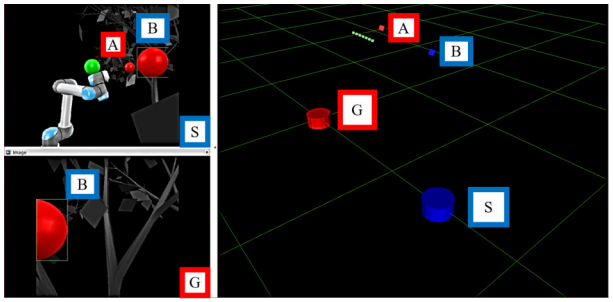

Fig. 9: Camera feeds (left) and graph building (right).

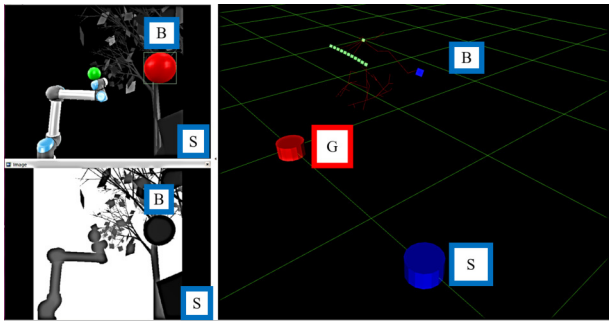

Fig. 10: Camera (top-left) and depth (bottom-left) feeds from S and generated RRTs (right).

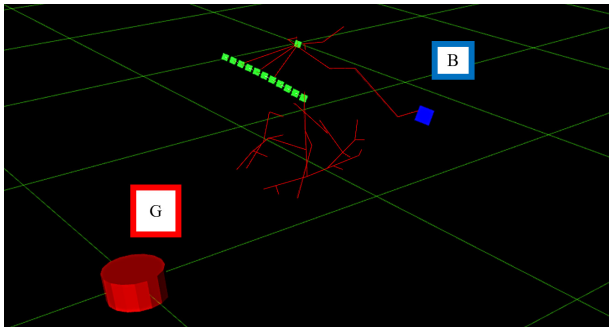

Fig. 11: RRT generated using S's depth feed.

poses. Shown in Fig. 8 is the example scenario discussed in this section. The two apples in the scene are labelled A and B. The grasp arm G and search arm S are labelled as per their abbreviations. This labelling convention is used for the remainder of this section. Fig. 8 indicates the state of the scene when the simulation is initialized; G positions itself looking forward toward the tree, beginning its predetermined scan routine as S is also seen beginning its scan whilst facing G. G's scan routine for this experiment entails pitching its camera up and down while translating toward S in small steps.

In Fig. 9, the panel images to the left have been recorded from a feed from S's (top) and G's (bottom) cameras during the simulation. The image to the right depicts what the graph records; the base of G and S are shown in red and blue respectively, apple B (blue square) has been discovered by S but has not been seen by G whereas A (red square) has been registered and added into $\mathcal{C}$ by G. The green squares represent positions occupied by G at different times during the simulation thus far.

After G finishes its scan, it uses $\mathcal{C}$ to approach and "grasp" apple A. It then performs a "drop" maneuver and confirms that no apples remain at the location previously occupied by A. This node is turned green. The search arm uses HPIRC to search and produce a path using the depth map. This is shown in Fig. 10; the image in the top left is a snapshot of the RGB feed supplied to S and the bottom left is that of the depth feed. The image to the right shows the newly generated tree that connects the blue node B to $\mathcal{C}$. This is shown clearly in Fig. 11.

B. Experiment

The experimental setup includes two UR5 arms (6-DOF) each equipped with an Intel Realsense D435 RGB-D camera in an eye-in-hand configuration. Artificial apple trees are used to simulate an agricultural setting. Apples are detected using a custom-trained version of deep learning architecture YOLO [11] and localized using depth readings from the cameras. The apple detection and localization pipeline is elaborated in [12].

The environment in Fig. 12 follows a set-up similar to that shown in the simulation example; only apples A and B are visible to G during its scan whereas apples C and D are S's responsibility. Following the initial scan, Fig. 13 shows apples A and B (red) have been discovered and registered by G and, C and D (blue) by S. G uses Dijkstra's algorithm to generate a path to A and B, and "picks" these apples one

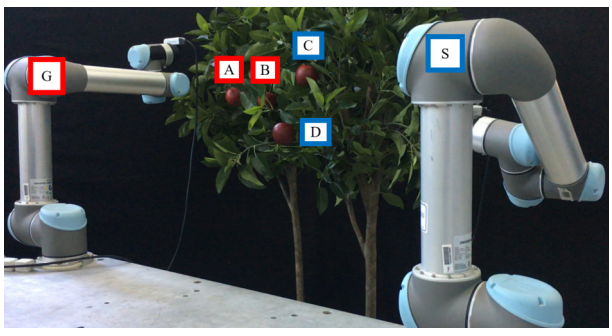

Fig. 12: Experimental setup in laboratory.

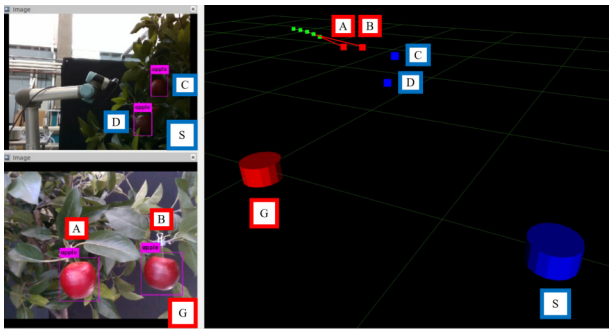

Fig. 13: RGB feeds (left) and graph after initial scan (right).

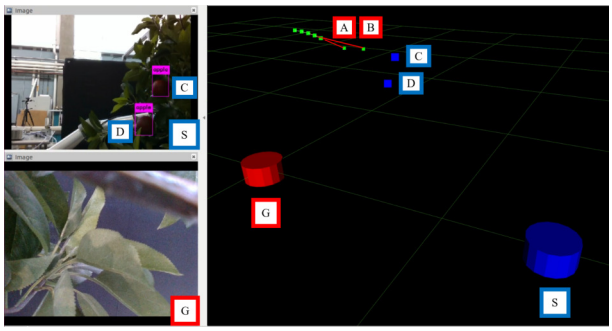

Fig. 14: RGB feeds (left) and graph after A and B are picked (right).

after the other. These apples are removed from the scene to allow G to turn the corresponding nodes green (Fig. 14), leaving C and D to be picked. The top-left and bottom left images in Fig. 15 show the RGB and aligned depth snapshots to S. HPIRC is able to leverage the depth map and expand the graph to connect previously unconnected nodes to $\mathcal{C}$. The paths to C and D account for the branches that prevent G from discovering these apples during the initial scan and force G to maneuver *around* them to reach the apples.

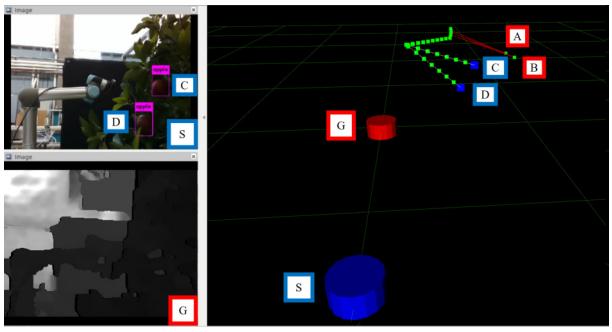

Fig. 15: RGB (G), Depth (S) feeds (left) and graph updated by HPIRC using S's depth image (right).

IV. CONCLUSIONS

This work focuses on solving the problem of path planning in unstructured agricultural settings using collaborative robotic arms that build and maintain a graph-based representation of targets and viable paths with no explicit recording or characterization of obstacles in the scene. This approach limits existing popular sampling-based path generation techniques to be applied directly. We bridge this limitation with HPIRC that deploys a greedy variant of RRT-Connect over sections of the observed scene, enabling the generation of paths in increments. Preliminary tests indicate that the concept is simple to implement, and performs reasonably well without the addition of more sophisticated heuristics and optimization. Further testing is required to collect quantitative performance data (success rate, computation time, path cost) and experiment with various problem settings and path optimizers.

ACKNOWLEDGMENT

This work is sponsored by the United States Department of Agriculture under a National Robotics Initiative grant.

REFERENCES

[1] K. J. Ahlin, A.-P. Hu, and N. Sadegh. Apple picking using dual robot arms operating within an unknown tree. In *2017 ASABE Annual International Meeting*, page 1. American Society of Agricultural and Biological Engineers, 2017.

[2] X. Cao, X. Zou, C. Jia, M. Chen, and Z. Zeng. Rrt-based path planning for an intelligent litchi-picking manipulator. *Computers and Electronics in Agriculture*, 156:105–118, 2019.

[3] J. R. Davidson, C. J. Hohimer, C. Mo, and M. Karkee. Dual robot coordination for apple harvesting. In *2017 ASABE Annual International Meeting*, page 1. American Society of Agricultural and Biological Engineers, 2017.

[4] Z. De-An, L. Jidong, J. Wei, Z. Ying, and C. Yu. Design and control of an apple harvesting robot. *Biosystems engineering*, 110(2):112–122, 2011.

[5] J. J. Kuffner Jr and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *ICRA*, volume 2, 2000.

[6] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[7] C. F. Lehnert, A. English, C. McCool, A. W. Tow, and T. Perez. Autonomous sweet pepper harvesting for protected cropping systems. *IEEE Robotics and Automation Letters*, 2(2):872–879, 2017.

[8] T. T. Nguyen, E. Kayacan, J. De Baedemaeker, and W. Saeys. Task and motion planning for apple harvesting robot. *IFAC Proceedings Volumes*, 46(18):247–252, 2013.

[9] N. Noguchi, J. Will, J. Reid, and Q. Zhang. Development of a master–slave robot system for farm operations. *Computers and Electronics in agriculture*, 44(1):1–19, 2004.

[10] S. Paulin, T. Botterill, J. Lin, X. Chen, and R. Green. A comparison of sampling-based path planners for a grape vine pruning robot arm. In *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, pages 98–103. IEEE, 2015.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[12] H. Sarabu, K. J. Ahlin, and A.-P. Hu. Leveraging deep learning and rgb-d cameras for cooperative apple-picking robot arms. In *2019 ASABE Annual International Meeting*, page 1. American Society of Agricultural and Biological Engineers, 2019.

[13] Y. Sarig. Robotics of fruit harvesting: A state-of-the-art review. *Journal of agricultural engineering research*, 54(4):265–280, 1993.

[14] R. R. Shamshiri, C. Weltzien, I. A. Hameed, I. J. Yule, T. E. Grift, S. K. Balasundram, L. Pitonakova, D. Ahmad, and G. Chowdhary. Research and development in agricultural robotics: A perspective of digital farming. *International Journal of Agricultural and Biological Engineering*, 11(4):1–14, 2018.

[15] 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*, 127:311–323, 2016.

[16] B. Zion, M. Mann, D. Levin, A. Shilo, D. Rubinstein, and I. Shmulevich. Harvest-order planning for a multiarm robotic harvester. *Computers and Electronics in Agriculture*, 103:75–81, 2014.