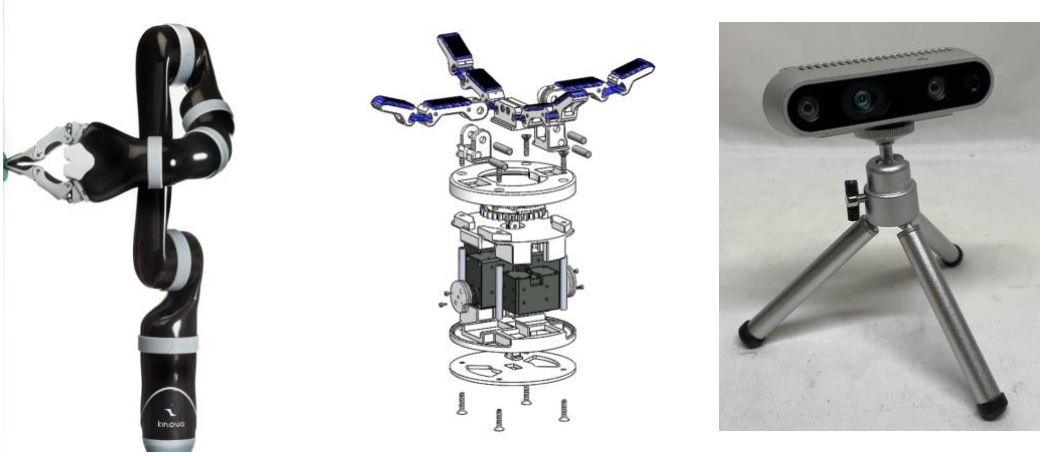
Project report: Robotic fruit grasping

Abstract: In this project, we focus on a ROS-based robotic manipulator control and perform a sweet pepper grasping task, including the gazebo simulation and real situation. A commercially available robotic arm was equipped an RGB-D camera used to detect a correct position to grasp peppers. Specifically, we trained and applied lightweight YOLOv4-tiny model for sweet pepper detection in 2D image. And then, the detected ROIs were mapped into the 3D point cloud for determining real 3D position. Then the robotic arm was controlled to approach the fruit based on the kinematics and MoveIt software. Overall, we finished part of the project objectives and need more time to further develop it.

Hardware:

- 6 DoF robotic arm: Kinova mico2
- Gripper: Commercial gripper or Self- customized 4 DOF tendon-driven gripper
- RGB-D camera: RealSense D435i
- Laptop: Ubuntu 20.04, ROS noetic, gazebo11, MoveIt1



(a) 6 DoF Robotic arms (b) Customized gripper (c) RGB-D camera

Figure1: Hardware components of fruit grasping system

Specifically, the coordinate system definition of robotic arm and the classic DH table can be known and calculated from the Kinova manual, as shown as the figure 2.

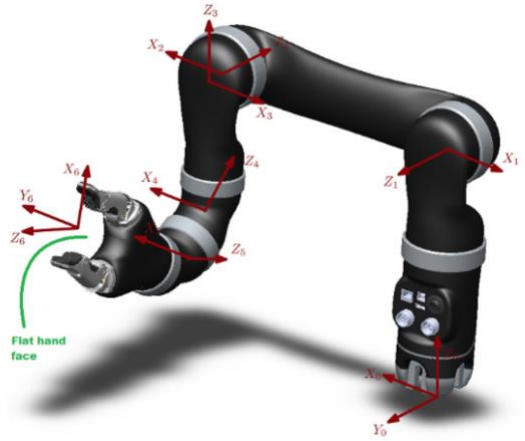


Table 1. Classic D-H table of MICO2 robotic arm

Joint #	d_i (m)	a_{i-1} (m)	α_{i-1}(rad)	θ_i (rad)
1	0.2755	0	$\pi/2$	θ_1
2	0	0.2900	π	$\theta 2$
3	0	-0.0741	$\pi/2$	θ_3
4	0.1661	0.1663	$\pi/3$	θ_4
5	0.856	0.08614	$\pi/3$	θ_5
6	0.2028	0.203	π	θ_6

Figure 2: Kinematics parameters of robotic arm

Considering the end-effector is controlled with an open-close mode, the transformation matric

of fingers are not shown. We plan to use two different grippers to achieve the fruit grasping: one is the Kinova two-finger gripper, and another is the customized tendon-driven gripper.

System Control Architecture diagram:

The state machine is central to the fruit grasping system and contains the decision making logic. Figure 3 illustrated the system control architecture diagram. The RealSense registration script can capture RGB and depth images, and then further register 3D point cloud for scene. The fruit segmentation, grasp selection nodes and motion planning nodes are used to generate information required to harvest each fruit within a scanned scene. Physical harvesting operations are handled by the path planner, robot arm controller and end effector control subsystems.

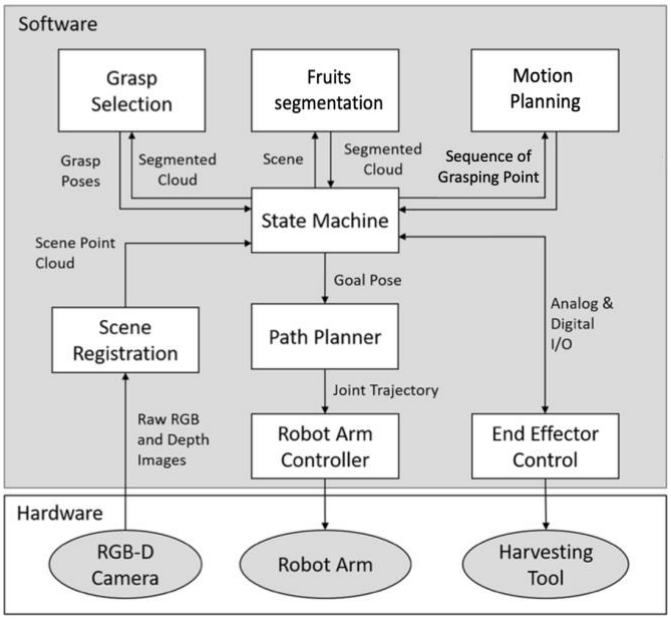


Figure 3: System Control Architecture diagram.

The following bullet points shows the main technologies we used during the experiment:

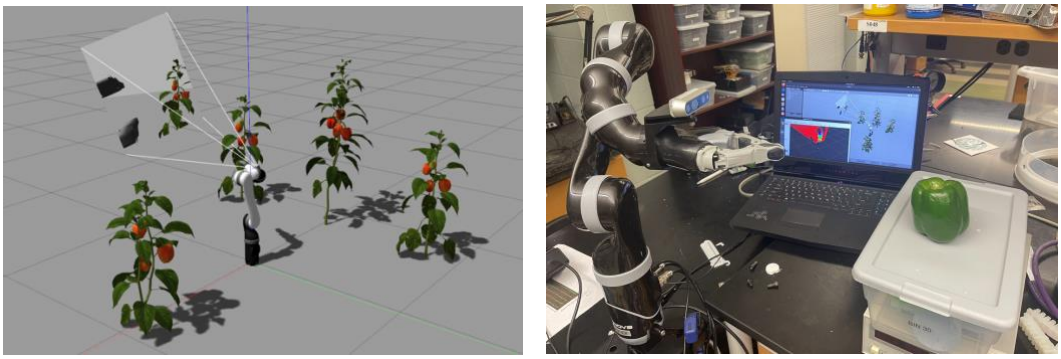
- **System control**: ROS noetic
- **Perception:** YOLOv4 based fruit detector, OpenCV-based image processing, PCL based point cloud processing
- **Manipulator control:** ROS MoveIt, OMPL controller
- **Motion planning for multiple objectives:** RRT algorithm
- **Hand-eye calibration**: visp_hand2eye_calibration tool (ROS package)
- **Simulator:** Gazebo

Experiment setting

We set specific scenarios for the fruit grasping test including gazebo simulation (figure 4a) and real world (figure 4b), respectively. A RealSense D435i camera is attached on the end of the robotic arm (joint 6). In the gazebo simulation, we set its transform matrix with the translation parameters (0.03, 0, -0.05) and the rotational parameters (0, pi/2, 0) from the joint 6. For the real world, the parameters should be calibrated with visp_hand2eye_calibration tool (http://wiki.ros.org/visp_hand2eye_calibration) to get more accurate transformation matrix.

In the simulation scenario, we set four sweet pepper plants around robotic arm and each plants

have different fruits and leaves. Those models are from an open-source GitHub project (https://github.com/Eruvae/ur_with_cam_gazebo) and I modified the plants to increase its realistic features and variety. As for the real world, it's hard to grow a pepper plant with multiple fruits at this time and we only use one single pepper without other leaves and trunks for the final test.



(a) (b)

Figure 4. Scenario setting for fruit grasping: (a) in gazebo simulation. (b) real world

Experiment and Result

2D Detection: A common approach in fruit detection is YOLOV4 deep learning model, based on a convolutional neural network (CNN) structure and Darknet architecture. Its lightweight version named YOLOV4-tiny are capable of fast simultaneous detection of multiple objects in a single shot. Darknet_ros (https://github.com/leggedrobotics/darknet_ros) is used to deploy the YOLO model in ROS.

We build a small dataset for the sweet pepper detection with total 80 RGB images, including 40 images from gazebo simulation and 40 real images. Transfer learning was used to increase the learning efficiency and generalization as the training dataset was small. In this study, the YOLOv4-tiny model was initialized by weights pretrained on MS COCO dataset (https://cocodataset.org/), and then fine-tuned on our training set.

These images contained three types of objects: green pepper, yellow pepper, and red pepper. We annotated the images with an online annotation tool (Roboflow) and then augmented them to 10x images with rotations, flips, shears, and exposures. The dataset was divided into three sections, including a training set, a testing set, and a valid dataset with the proportion of 7:2:1. Table. 1 shows the best performance of the trained weights. Because of the relative simplicity of the detected objects and background, the average accuracy rose to 98.46%.

Considering the dataset is too small, the detection accuracy in real situation cannot achieve so high. In this project, we focus on the entire pipeline development and it can be further improve with more extensive dataset.

Table 2. Training Performance of YOLOv4-tiny model for cotton plants and weeds detection

	Class	AP	Recall	F1 score	mAP@0.5
YOLOv4-tiny model	Green pepper	97.98%	0.98	0.97	98.46%
	yellow pepper	96.83%			
	red pepper	98.94%			

(a) (b)

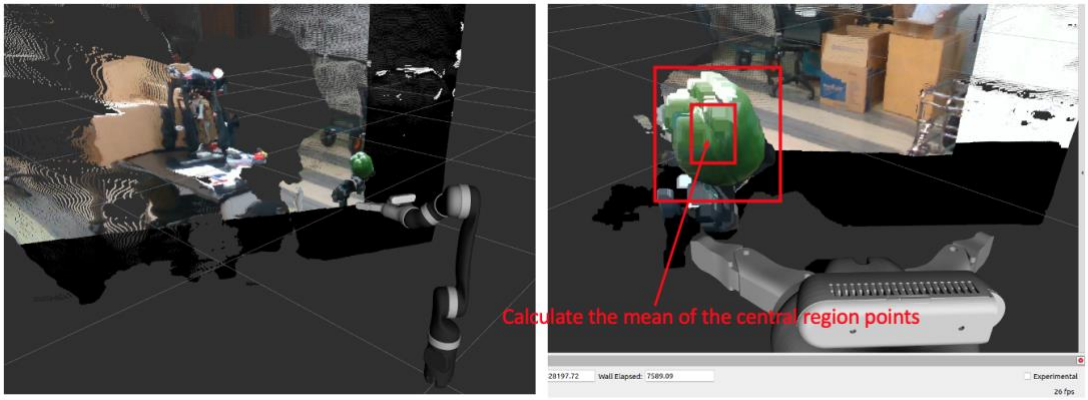
Figure 4. The detection result of YOLO-based detector

2D-3D mapping： Generally, the intrinsic and extrinsic of cameras are always used to calculate the point cloud (x, y, z) from the aligned depth and RGB images. Through launching RealSense registration node, the point could be subscribed with ROS topic subscriber.

I observed the dimension of the point cloud equal to the number of the RGB image pixels. In our case, both the resolution of aligned depth image and RGB image are 640*480, and the number of point cloud is 307200. That means for each pixel in 2D image we can index its 3D coordinate (x,y,z) (relative to camera coordinate system) according its pixel location. Given the pixel location (u,v), its corresponding point cloud is (640*u+v).

With this corresponding relationship, the point clouds of the detected bounding boxes can be indexed. I use a color filter to remove the non-green pixels, and the depth filters to remove those point clouds too far away from the camera (more than 2.5 miters).

In order to calculate the 3D location of the pepper, A scaled region is selected to overcome noise influence caused by the background. Specifically, we calculate the average (x, y, z) value of the central region with the half-length and half-width as the pepper's location.



(a) (b)

Figure 5. Result of 3D mapping. (a) Point clouds and the robot model in RVIZ; (b) illustrates that we only calculate the region of the central scaled bounding box

Pose Extraction: In our project, the pepper's pose should be fitted as a cylinder with PCL API. Then the pose can be extracted from the cylinder's pose. We face some difficulty to achieve this section. Finally, we decided to send a fixed pose of the pepper, which means the gripper pose will always keep a specific value without change.

Manipulator Control: Kinova provides a Ros package to drive the robotic arm and gives some examples to control the robotic arm: https://github.com/Kinovarobotics/kinova-ros.
We used the cartesian position control mode to control the end-effector approach the pepper. Before that, the desired 6D pose are required to be transformed to the robot base coordinate from the camera coordinate. PCL provide the transform API to achieve it.
Then, the kinova_tool_pose_action (action server called by pose_action_client.py) will send Cartesian position commands to the robot and the inverse kinematics will be handled within the robot. Important The inverse kinematics algorithm that is implemented within Kinova robots is programmed to automatically avoid singularities and self-collisions.

Customized 4DoF gripper: In order to grasp the pepper successfully, it requires quite accurate estimation of fruit location and hand-eye calibration parameters. Besides, considering the fruit's size, the radius estimation is also pretty important: too large estimation cause the gripper cannot hold the fruit; and too small estimation will damage the fruit.
In this work, we tried to control two types of grippers and compared the grasping successful rate and fruit damage rate after grasping. However, the commercial gripper has a hardware issue and cannot communicate with laptop. I made an effort to fix it, for example, change the damaged communication cable. But it seems like a short circuit issue.
Besides, as for the tendon-driven gripper, it hasn't finished so far. I have fabricated most of the gripper and will use four servo motor to achieve the action of grasp and release. I have written a ROS node to control single motor, but for the entire gripper, I need more time to realize it.

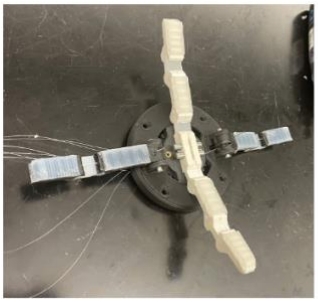


Figure 6. The detection result of YOLO-based detector

Experiment:
We test the algorithms to grasp a green pepper in real situation. Due to the broken commercial gripper and un-finished tendon-driven gripper, we only control the robotic arm to approach the fruit. The video can be seen in demo1_perception.mp4 and demo2_approach pepper with robotic arm.mp4.

Further work:

- Hand-eye calibration to get more accurate transform from camera to end-effector
- Cylinder fitting for the segmented point cloud should be developed
- End-effector fabrication and comparation
- Localization accuracy test for the end-effector
- Motion planning algorithm test: RRT for obstacle avoidance and A* for multiple fruits grasping