Object Detection and Depth Estimation using Stereo Camera for Pick & Place Operation using Custom-built Manipulator

Arshad Shaik
UID: 118438832
University of Maryland
arshad22@umd.edu

Dhinesh Rajasekaran
UID: 119400241
University of Maryland
dhinesh@umd.edu

Venkata Tej Kiran Reddy
UID: 119197066
University of Maryland
itej89@umd.edu

Abstract

In this project, a custom-built Robot Manipulator will be used for picking and placing of an auto-detected object, for which a stereo depth estimation is performed using 2 onboard imagers. We will calculate the world coordinates of an object in the camera scene or image and do coordinate transformation from camera frame to end effector frame. Then the manipulator would perform any simple action on the object such as lifting it or just moving the object.

1. Introduction

For depth estimation of an object for pick and place operation using a manipulator, we have used a hardware synchronized off-the shelf stereo camera available online. We have interfaced it a laptop directly using CSI to USB converter. Then using basic object detection pipeline using blob or edge detection (we will have only the target object in the camera scene), the object is identified, to be picked or moved and using concepts we learnt in epipolar geometry and stereo calibration, we will estimate the depth of the object w.r.t to camera frame. Once we have the 3D world coordinates w.r.t to camera frame, coordinate transformation is performed to convert the coordinates from camera frame to manipulator or end-effector's frame. Subsequently, the picking or moving action of the manipulator using the 3D coordinates provided. Other building blocks of the manipulator like inverse kinematics and path planning is handled by pre-built MoveIt packages. The manipulator is a 4 DoF with or without an end-effector.

2. Methodology:

The approach for the above problem is as below:
1. Get individual image frame from both the cameras and synchronize the frame sequence
2. Create a pipeline for stereo for depth estimation and point cloud creation.
3. Extract the required object's depth from the estimated point cloud.
4. Come up with the transformation between manipulator base frame and camera frame.
5. Identify a 3D point in the space for the manipulator to implement a pick and place operation.

Overall – Methodology:

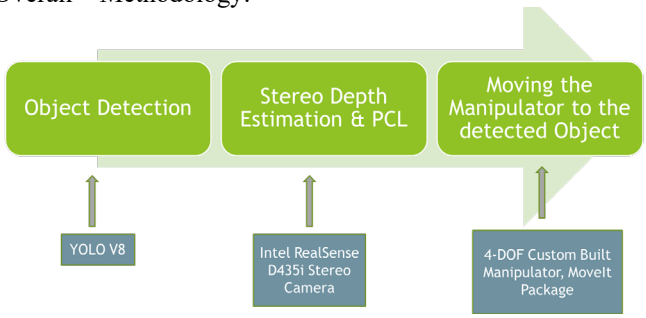


Figure 1: Overall Methodology

2.1. Stereo Camera:

Intel® RealSense™ depth camera D435i combines the robust depth sensing capabilities of the D435 with the addition of an inertial measurement unit (IMU).

It is ideal for rudimentary SLAM and tracking applications allowing better point-cloud alignment.

Figure 2: Intel Realsense Stereo Depth Camera D435i

When using the D435i, Intel RealSense SDK 2.0 provides IMU data that is time stamped to align with our high-quality depth data.

Stereo Camera Tech Specifications:

Table 3-11. Stereo Depth Module SKU Properties

Stereo Module	Intel® RealSense™ Depth Module D410	Intel® RealSense™ Depth Module D415	Intel® RealSense™ Depth Module D430
Baseline	55 mm	55 mm	50 mm
Left/Right Imagers Type	Standard	Standard	Wide
Depth FOV HD (16:9) (degrees)	H:65 / V:40 / D:72	H:65 / V:40 / D:72	H:87 / V:58 / D:95
Depth FOV VGA (4:3) (degrees)	H:50 / V:40 / D:61	H:50 / V:40 / D:61	H:75 / V:62 / D:89
IR Projector	Standard	Standard	Wide
IR Projector FOV	H:67 / V:41 / D:75	H:67 / V:41 / D:75	H:90 / V:63 / D:99
Color Sensor	-	OV2740	-
Color Camera FOV	-	H:69/ V:42 / D:77	-
Module Dimensions (mm)	X=74.7 mm Y=10 mm Z=4.7 mm	X=83.7 mm Y=10 mm Z=4.7 mm	X=70.7 mm Y=14 mm Z=10.53 mm

Table 3-13. Wide Left and Right Imager Properties – D430

Parameter	Camera Sensor Properties
Image Sensor	OmniVision Technologies OV9282
Active Pixels	1280 x 800
Sensor Aspect Ratio	8:5
Format	10-bit RAW
F Number	f/2.0
Focal Length	1.93 mm
Filter Type	None
Focus	Fixed
Shutter Type	Global Shutter

Parameter	Camera Sensor Properties
Signal Interface	MIPI CSI-2, 2X Lanes
Horizontal Field of View	90° ± 1°
Vertical Field of View	64° ± 1°
Diagonal Field of View	98 ± 1°
Distortion	<= 1.5%

Table 3-18. Color Sensor Properties – D415

Parameter	Camera Sensor Properties
Image Sensor	OmniVision Technologies OV2740
Color Image Signal Processor	Discrete
Active Pixels	1920 x 1080
Sensor Aspect Ratio	16:9
Format	10-bit RAW RGB
F Number	f/2.0
Focal Length	1.88 mm
Filter Type	IR Cut Filter
Focus	Fixed
Shutter Type	Rolling Shutter
Signal Interface	MIPI CSI-2, 1 Lane
Horizontal Field of View	69.4°
Vertical Field of View	42.5°
Diagonal Field of View	77°
Distortion	<= 1.5%

2.2. Object Detection – Pipeline:

A pre-trained neural network model – YOLO V8 – is used to detect an object (in this case, object is water bottle).

The object detection pipeline is outlined as below:

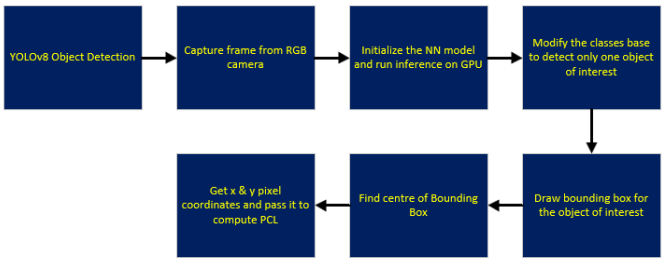


Figure 3: Object Detection - Pipeline

2.3. Depth Estimation - Pipeline:

Pipeline for the detection of object's real world co-ordinates is implemented as follows.

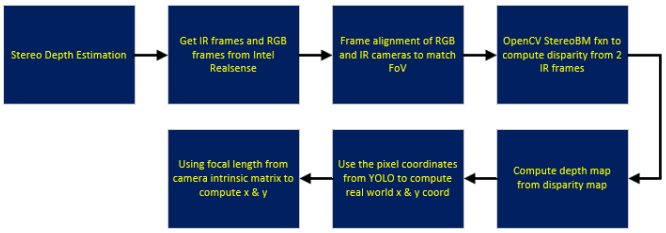


Figure 4: Depth Estimation - Pipeline

Realsense 435i has an RGB sensor, two infrared cameras and one IR projector. IR projector projects a pattern onto the scene to make sure there are features that can be extracted to compare the two IR camera feeds.

As the first block of our software stack, we have processed the RGB information using the Yolov8 neural network to detect the object of interest. We have modified the V8 output to detect a single object, which in our case is a water bottle.

The output of Yolov8 network is the coordinates of the bounding box drawn around the detected object. From the coordinates of the bounding box corners, we will compute the location of the center of mass of the object at the intersection of diagonals.

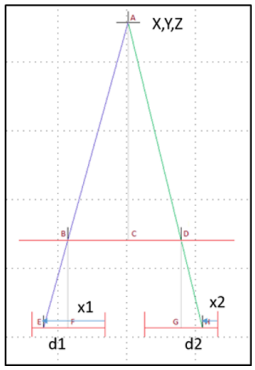
Next, we will read the IR stereo images from the Realsense camera. We made sure that the IR images are read at the same resolution settings as RGB images. Then we passed these images through a Realsense image align API to align the pixel coordinates with the RGB camera feed.

The aligned stereo images are passed through OpenCV BF matcher's stereo API to compute the disparity map.

The disparity map is converted to real world depth map using the formula.

Depth = baseline * focal length / disparity

Triangulation



$$\frac{x_1 - 0.5 * d_1}{f} = \frac{0.5 * T}{Z}$$

$$\frac{0.5 * d_2 - x_2}{f} = \frac{0.5 * T}{Z}$$

$$\frac{x_1 - x_2}{f} = \frac{T}{Z}$$

If both cameras have the same size sensor

45

To compute the x and y coordinates of the object center of mass, first we will extract the depth of the respective pixel from the depth map.

To use the similarity triangle technique, we need to find the object center of mass location in millimeters from the pixel coordinates. This can be done by computing the degree per pixel from the camera's vertical and horizontal field of view using the below formula.

Objects center of mass X in mm = (Xpx - width/2) * (horizontal_deg/pixel)
Objects center of mass Y in mm = (Ypx - height/2) * (vertical_deg/pixel)

where,

horizontal_deg/pixel = horizontal _FOV / Image Width in pixels
vertical_deg/pixel = vertical_FOV / Image Height in pixels

Once we extract the CoM, X, Y image coordinates in mm, we can use similar triangle technique to extract the real world x, y coordinates using the below formulas
X_world = Xmm/Focal_length * depth
Y_world = Ymm/Focal_length * depth

The computed X, Y, Z world coordinates are forwarded to MoveIt to perform the goal reaching task
using the manipulator.

The IR images from both the left and right IT imagers, and the RGB image are shown below.

The resultant disparity map of the respective image is also shown here.

2.3.1 Current Left and Right IR Camera Feeds:

Figure 5: Left and Right IR Image Feeds

2.3.2 RGB Camera Feed:

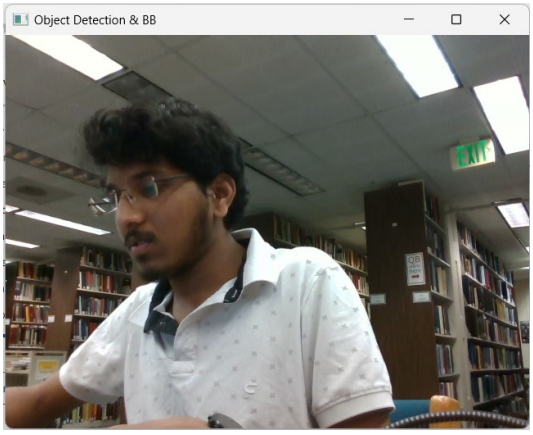


Figure 6: RGB Camera Feed

2.3.3 Disparity Map:

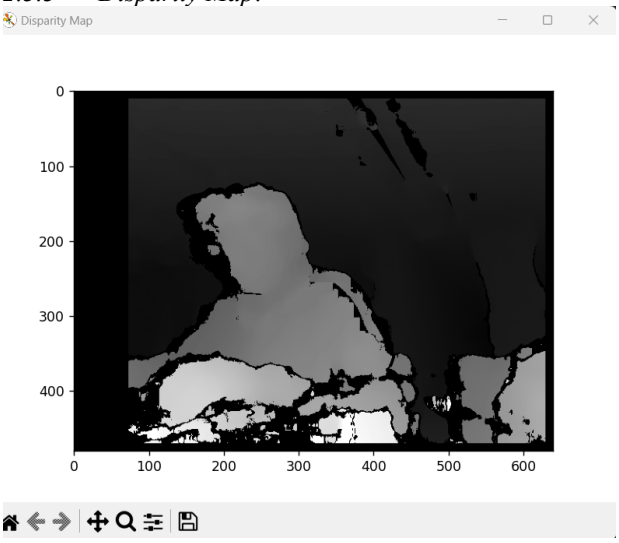


Figure 7: Disparity Map

2.4. Custom-built Manipulator:

We have designed and built a manipulator from scratch as part of our Planning course. We want to integrate our

object location detection algorithm with this manipulator to perform pick and place operations.

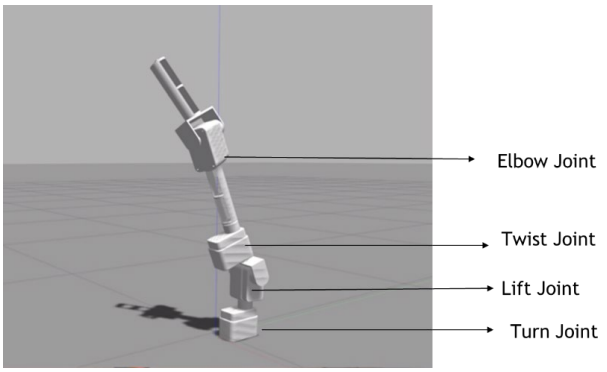


Figure 8: Custom-built Robot Arm Manipulator

To achieve this, we have made our robot compatible with the ROS MoveIt framework. We have developed a complete software stack to make the manipulator follow the trajectory created by the ROS RRT planner. We have created interfaces to send goal location to this planner. We have verified that the planner takes the goal location and is able to create a series of joint angles to be followed by the manipulator to reach this goal location.

We have verified this functionality by providing MoveIt with a real-world goal and recording the path followed by real manipulator matches with the simulation in Gazebo and RviZ.

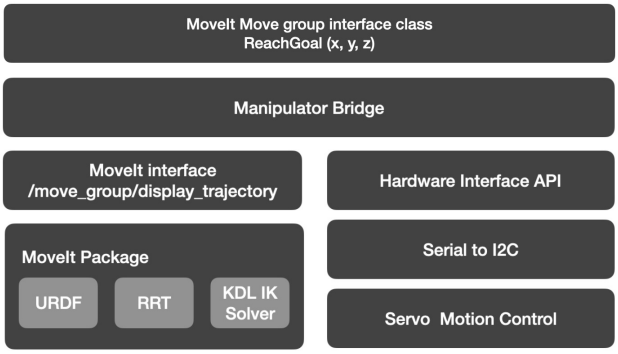


Figure 9: Manipulator Software Architecture

3. Results:

3.1. Demo Video 1:

YOLOv8 Object Detection and Stereo Depth Estimation using Intel Realsense 435i IR camera feed. Validated using measuring tape.

https://www.youtube.com/watch?v=uZFN3-d_hQE

3.2. Demo Video 2:

Moving the custom-built manipulator to target 3D coordinate using MoveIt package developed and visualized using RVIZ, Gazebo and on Real Hardware.

https://www.youtube.com/watch?v=7Xjhg34qU5E

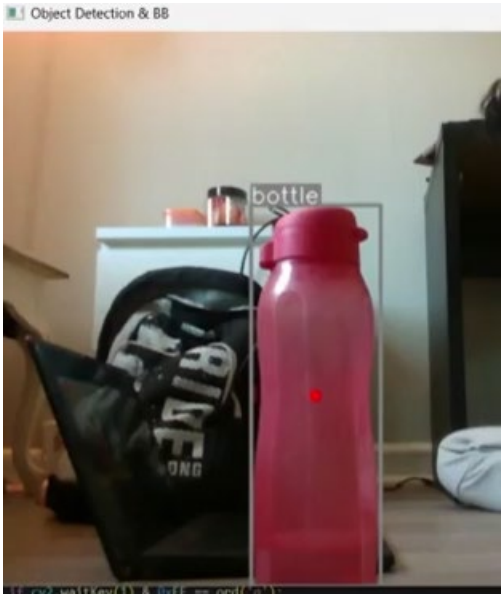


Figure 10: Object Detection by YOLO V8

3.3. How the results may be made better:

a. It needs to be further investigated on why the depth estimations are not as good as the one that is provided by on device stereo camera depth.
b. Co-ordinate transformation between the camera and the manipulator pipeline block needs to be implemented.
c. Gripper needs to be installed for the manipulator for the complete autonomous operation of manipulator pick and place operation
d. Requires mechanical changes to make the manipulator structurally stable.

4. Challenges Faced:

a. Camera Interface Issue: When we connected the camera using the USB interface to the laptop, no live video feed was received. After some exploration, it is understood that it requires a USB 3.1 or 3.2 cable to work properly.
b. Electronic Boards: Few of the electronic boards were burnt, due to the misconnection of one connector with the other.
c. Depth estimation using own-stereo pipeline was not precise.

d. Estimation of x and y co-ordinates was initially incorrect because of misalignment of frame size from IR cameras and RGB camera.
e. Manipulator servo joint especially the lift motor joint was weak and caused wobbling while trying to move. Additional L-angle is attached for strengthening.
f. Getting 3D point cloud and verifying their accuracy.

5. Contributions:

Project as a Whole:
a. Auto-detection of an object and movement of a robotic arm to the object
b. Implementation of Depth Estimation Pipeline using individual image frames from the two IR imagers of the camera
c. Hardware-based Project (Stereo Camera, Robotic Arm)

Individual Contributions:
a. Arshad: Auto-detection of an object using YOLO v8
b. Dhinesh: Implementation of Stereo Depth Estimation Pipeline
c. Tej: Integration of Perception Stack and Manipulator Software stack

6. Conclusion:

In this project,
- Object detection is implemented using YOLOv8
- Depth estimation pipeline (own & built-in) is implemented to get a 3D point for the manipulator operation.
- Demonstrated the movement of manipulator to the desired location both in simulation and on real hardware.
- Our project integrates perception, planning and modelling into single pipeline and demonstrates in real hardware.

Future Work:
- More accurate depth estimation
- Installation of Gripper

7. Acknowledgements:

We would like to acknowledge Prof. Samer Charifa and the course Tas – Arunava Basu, Madhu Chittibabu, Kumar Oruganti, for their guidance and support in the execution of this project.

8. References:

[1] https://learnOpenCV.com/depth-perception-using-stereo-camera-python-c/
[2] https://medium.com/analytics-vidhya/distance-estimation-cf2f2fd709d8
[3] https://moveit.ros.org/