Improving Robotic Perception by Refining 6D Object Pose Estimation using Push Based Manipulation

An RGB-D and Point Cloud Approach Integrating Object Manipulation with Pose Estimation

Zak Quor

A report submitted in partial fulfilment of the requirements for the degree of Masters of Robotics and Automation, The University of Auckland, 2024.

Abstract

This research investigates improvements in robotic perception for enhanced 6D object pose estimation using push-based manipulation in conjunction with a stereo depth sensor and a robotic arm. Through this study, the application of RGB-D data, derived from the Intel RealSense D455, is explored alongside a Universal Robots UR5e cobot for tasks involving object identification, orientation assessment, and physical manipulation trajectory planning. Utilising the Point Cloud Library (PCL) and the Viewpoint Feature Histogram (VFH) descriptor, this project aims to develop an approach that balances real-time performance with high accuracy in object pose estimation, leveraging the strengths of depth-based data over RGB alone.

In reviewing existing 6DoF pose estimation techniques—covering template-based, feature-based, and learning-based methods—this study identifies the computational efficiencies and adaptability of VFH-based feature extraction as essential for handling occlusions and refining pose estimates in dynamic environments. RGB-D data is shown to improve depth perception, providing the necessary spatial data for calculating approach vectors critical in push-based manipulation. Through iterative testing, the system demonstrated high accuracy in maintaining a consistent robot-object alignment, showing potential for integration into more complex and cluttered scenes where object geometries and poses frequently shift.

This research opens up opportunities for future work to focus on enhancing pose estimation reliability by integrating sensor feedback and refining the cluster extraction algorithm for diverse object recognition. Additional objectives could include implementing a ROS-based system for improved interoperability and exploring sensor fusion with multi-camera setups to increase pose estimation precision. This research contributes to robotic vision by advancing techniques for real-time 6D pose estimation, offering a pathway to improved robotic perception and interaction with the physical world.

Acknowledgements

I would like to express my special thanks to my current employer who considerately allowed me to take extended leave to focus on conducting this research project. I may not have had the capacity to complete this research without their generosity. I would also like to express my gratitude to my project supervisor and the postgraduate research team in the Mechatronics Laboratory at The University of Auckland for their much appreciated guidance.

Table of Contents

Abstract 3
Acknowledgements 4
List of Figures 6
List of Tables 7
List of Abbreviations 8
Section 1. Introduction 9
Section 2. Related Work 11
2.1. 6D Pose Estimation Methods 11
2.2. Balancing RGB, Depth, and RGB-D for Real-Time Performance 13
Section 3. Experiment Method 15
3.1. Goal 15
3.2. The Test Environment 15
3.3. Designing the software 17
3.4. Analysing the system behaviour 18
3.5. Testing method 22
Section 4. Results 23
4.1. Presentation of results 23
4.2. Error 26
4.3. Discussion 27
Section 5. Conclusions 29
5.1. Summary 29
5.2. Future work 30
References 32
Appendices 33
1. Measurement data collected from experiments 33
2. Universal Robotics UR5e Cobot Datasheet 41
3. Intel Realsense Depth Camera D455 Technical Specifications 42

List of Figures

Figure 1. An RGB, category-level pose estimation technique named Center Pose 12

Figure 2. Labelled photograph of the laboratory testing scene and equipment. 16

Figure 3. D455 Camera and End Effector attachment. 16

Figure 4. Objects used during the experiment process. 16

Figure 5. Depth stream of 2 cubes in scene. 18

Figure 6. Depth stream of 2 cubes in rotated position. 18

Figure 7. Cubes in a scene viewed from two different vertical angles 19

Figure 8. Intel Realsense D455 lens and emitter diagram. 20

Figure 9. Depth image demonstrating offset and coordinate system of camera. 21

Figure 10. Spray Bottle VFH signature. 21

Figure 11. Spray Bottle point cloud. 21

Figure 12. Photograph of scene with spray bottle object and TCP ready for testing. 22

Figure 13. Diagram of the test scene, including 4 test positions, camera, TCP and w.r.f.. 23

Figure 14. Graph of Intel Realsense D415 and D435 Depth RMSE over Distance. 26

List of Tables

Table 1. Test Results: Centred, 400mm, 10 samples 23

Table 2. Test Results: Off-centre, 400mm, 10 samples 23

Table 3. Test Results: Off-centre, 600mm, 10 samples 24

Table 4. Test Results: Centred, 600mm, 10 samples 24

Table 5. Average absolute difference between actual and estimate 27

Table 6. Standard deviation including error for each test location 27

List of Abbreviations

Abbreviation	Definition
6DoF	6 degrees of freedom
RGB-D	Red, Green, Blue and Depth
VFH	Viewpoint Feature Histogram
LIDAR	Light Detection and Ranging
PCL	Point Cloud Library
TCP	Tool Centre Point
W.r.f	World Reference Frame
RMS(E)	Root Mean Squared (Error)

Section 1. Introduction

Robotic perception is a critical aspect of autonomous systems, enabling robots to understand and interact with their environment. A key challenge in this domain is achieving accurate 6 degrees of freedom (6DoF) pose estimation of objects, which involves determining an object's position and orientation in three-dimensional space. The ability to precisely estimate 6DoF poses is vital for applications such as object manipulation, navigation, and human-robot interaction. Despite significant advancements in pose estimation techniques, many existing algorithms struggle with accuracy, particularly in cluttered environments or when dealing with occluded (obscured) objects. This problem is especially prominent in vision-based methods which rely on camera data to infer the spatial properties of objects.

Existing 6DoF pose estimation algorithms have shown substantial advancements, yet they continue to face limitations in accuracy and reliability. Many traditional approaches fail in real-world settings where variability in lighting, object reflectivity, and background complexity introduces significant disruption to pose estimation. For example, even state-of-the-art vision-based methods, which utilise RGB-D data streams, often lack robustness against these variables, leading to inconsistent and sometimes unreliable pose estimations. These limitations create barriers for applications where consistent and precise interaction with objects is crucial, such as industrial robotics and assistive technologies. Therefore, refining robotic perception and making it robust against these environmental challenges is essential for advancing practical applications.

To address these limitations, this research project explores the integration of push-based manipulation with 6D pose estimation using extrinsic dexterity. The hypothesis is that by leveraging a robot's physical interaction capabilities, such as grasping and pushing objects to reveal more information, the accuracy of 6DoF pose estimation can be improved. This method combines visual feedback from RGB-D cameras with physical manipulators to iteratively refine object pose, providing a more precise perception of the object's spatial orientation.

The project focuses on incorporating forward-kinematic information of a robotic system's end effector, training models for everyday objects and utilising robot arm motion planning to update pose estimates through forward kinematics. This approach aims to enhance robotic

perception, enabling more accurate and reliable manipulation of objects in dynamic and uncertain environments. This project further explores how push-based manipulation and accurate 6DoF pose estimation can enable reliable object interaction.

The design of the software supporting this approach was designed to maximise the understanding between the perception and manipulation systems. Built and executed on Ubuntu 20.04, the system integrates libraries and dependencies from various sources to enable functionality across several C++ files. These files handle tasks such as processing the Intel RealSense D455 camera stream, extracting point clouds, and performing collision analysis. Key functions developed for the project include cluster extraction from point clouds, Viewpoint Feature Histogram (VFH) feature analysis, and the creation of a user-defined object library, allowing for the storage and quick retrieval of object profiles. By constructing a Kd-tree for fast referencing of new objects in the scene, the system improves its recognition time and handling of previously encountered objects. This efficiency is critical for real-time applications where a delay in pose estimation could result in operational inefficiencies, errors in manipulation, or failure.

In the physical test environment, certain object characteristics, lighting conditions, and orientations were evaluated to understand the influence of these variables on pose estimation accuracy. The RealSense D455 camera's depth perception for instance proved particularly sensitive to the surface properties of objects, with reflective surfaces like those on black spray bottles posing distinct challenges for recognition. Tests showed that less reflective, diffuse surfaces offered better shape definition in the depth stream, improving object detectability and subsequent pose estimation accuracy. This observation guided adjustments to the object identification and positioning algorithms to optimise detection and recognition conditions in real-world scenarios.

Additionally, the accuracy and limitations of both the robot and camera sensors were factored into this research to establish realistic expectations for pose estimation precision. The UR5e cobot, with a repeatability accuracy of ±0.03 mm, provided a stable and controlled reference for positioning objects during testing. Similarly, the D455 camera's depth accuracy, reported by Intel to be within ±2% at a 4 m range, was tested to understand its implications on x, y, and z axes' precision. These calibrations highlighted the need for an algorithm that could adapt to minor inaccuracies, especially in the x-axis, where variation due to factors like object reflectivity and noise was most significant.

Through a comparison of actual vs camera-estimated object poses, results indicated a high potential for this approach to improve pose estimation accuracy within tolerances suitable for push-based manipulation. Findings from test cases suggest that this method, despite minor deviations on certain axes, holds promise for enhancing robotic perception in uncertain or cluttered environments. Particularly, the project identified that depth readings on the z-axis were consistently within a 3-4 mm margin of error, proving sufficiently reliable for push-based adjustments. However, x-axis variability introduced by noise and interference with infrared sensors indicated areas for potential refinement, such as enhancing clustering algorithms or adjusting camera positioning.

The overall objective of this project was to push the boundaries of robotic perception by combining 6DoF pose estimation with tactile manipulation strategies, ultimately aiming to empower robots to move and interact within real-world environments with higher precision and adaptability. This approach represents a step toward equipping robots with an enhanced ability to understand their surroundings, allowing them to make estimates about objects even where traditional perception methods might lack the capacity.

Section 2. Related Work

2.1. 6D Pose Estimation Methods

The problem of 6DoF pose estimation has been widely studied, with a range of techniques developed over the years, each with its strengths and limitations. A comprehensive survey paper, “A Survey of 6DoF Object Pose Estimation Methods for Different Application Scenarios” [1], outlines in good detail various approaches tailored for specific needs. This paper describes how the methods can be broadly categorised into two groups: instance-level, which focuses on specific known objects, and category-level, which generalises across object categories. Within these groups, three primary types of data are used: RGB, point cloud (depth-based), and RGB-D data. Methods for pose estimation can be further classified into template-based methods, which rely on pre-stored object views; feature-based methods, which match keypoints between images and models; and learning-based methods, which use neural networks to predict poses. Each approach offers different levels of robustness and computational complexity, making them suitable for varying application scenarios.

Template-based methods match input images to a set of predefined models, but they often suffer from scalability issues due to the need for extensive model libraries. Feature-based methods, which rely on detecting and matching features between the 3D model and the scene, offer better scalability but struggle with both feature-rich and textureless objects. Learning-based methods provide end-to-end solutions capable of handling complex scenes and occlusions better than the other two methods. However, learning-based methods typically require large datasets and significant computational resources for training and inference.

Given a single RGB image containing previously unseen instances of known categories (in this case *cereal boxes*, *cups*, and *shoes*), our proposed method detects objects and estimates 6-DoF poses and 3D bounding box dimensions up to a scale factor. We use a separate network for each category.

Figure 1. An example of an RGB, category-level pose estimation technique named Center Pose[2]. Includes coordinate systems and bounding boxes for each object.

When considering these methods for real-time object manipulation and gripping, each has distinct advantages and limitations. Template-based methods can be effective for precise recognition of known objects but are less adaptable to unknown objects or those not represented in the model library. This makes them less suitable for dynamic environments where new objects might be encountered. Additionally, their reliance on pre-existing models can hinder real-time performance due to the need for extensive database searches.

Feature-based methods offer faster performance and better adaptability to varying object geometries, making them useful for real-time applications where computational efficiency is crucial. However, their reliance on identifiable features can be problematic when manipulating objects with few distinct surface features, such as smooth or reflective surfaces. This could result in a failure to accurately recognize objects as they are gripped and manipulated by the robotic arm, especially if parts of the object become occluded.

Learning-based methods, while powerful in terms of generalisation and handling occlusions, demand significant computational resources and training time, which can be impractical for

systems with limited hardware capabilities. Their ability to recognize new objects without extensive pre-modelling is an advantage for applications involving unknown objects. However, in real-time scenarios, the high inference times associated with deep learning models may introduce latency, potentially affecting the responsiveness of the robotic gripper during dynamic manipulation tasks.

2.2. Balancing RGB, Depth, and RGB-D for Real-Time Performance

The use of RGB, depth maps (or point cloud data), and RGB-D offers varying benefits and limitations to implementing real-time applications. RGB data alone provides detailed visual information, capturing texture and colour variations that can aid in object recognition. However, it lacks depth information which is essential for accurately estimating the spatial orientation and distance of objects in the scene. Depth maps or point clouds, generated from sensors like stereo cameras or LIDAR, provide detailed geometric structure, enabling precise 3D shape analysis. These methods, however, might struggle with low-texture surfaces or when the object's surface lacks sufficient variation for accurate depth sensing.

Using RGB-D data combines the strengths of both RGB imagery and depth maps, offering a better understanding of the scene for object manipulation tasks. The combination of data from these two methods allows for more accurate shape recognition and orientation estimation which is crucial when a robot needs to interact with objects of varying geometries, level of detail, and surface textures. For tasks involving gripping, depth information is particularly valuable as it helps the robotic arm gauge the precise distance and orientation of an object. Furthermore, the combined use of RGB-D data tends to be less computationally intensive than purely deep learning-based approaches, making it feasible for real-time scenarios. Although RGB-D methods might face challenges when dealing with entirely novel object categories, they can refine pose estimates through iterative manipulation, which is well-aligned with the requirements of robotic grasping.

An RGB-D approach that has garnered attention for real-time 6DoF pose estimation is the use of the Point Cloud Library (PCL)[3] with Viewpoint Feature Histogram[4]. PCL provides a comprehensive suite of tools for processing and analysing point clouds. VFH is a descriptor that encodes the shape and geometry of 3D point clouds, capturing information about an object’s surface normals and viewpoint.

In the context of this research project, PCL with VFH offers a compelling approach for integrating 6D pose estimation with robotic manipulation. The choice is motivated by several factors. Firstly, PCL's tools are well-suited for working with RGB-D data from depth sensors like the Intel RealSense. The Intel Realsense is able to stream both RGB and depth map data from the camera to descriptors like VFH. This allows for the relatively easy extraction of detailed 3D shape information which is crucial for accurately estimating the real-time object pose. There is easily accessible documentation on how to combine this Intel Realsense data with PCL[5].

Secondly, VFH-based recognition is effective in capturing the geometric properties of objects, providing a robust descriptor that is particularly valuable in scenarios involving occlusions[6]. This capability is critical for improving pose estimation accuracy during physical interactions, such as when a robotic gripper occludes parts of the object while grasping it. The depth data provided by RGB-D sensors allows the system to maintain awareness of the object's 3D structure, even when parts of the object become temporarily hidden. By utilising VFH, the system can adapt to changing visual information during manipulation and ideally continue to refine the pose estimate as the object's visibility changes throughout the grasping process.

Additionally, the real-time performance of PCL and comparatively low computing hardware requirements, makes it a practical choice for robotic systems that require timely feedback for manipulation tasks. This is particularly important in scenarios where the robot needs to adapt its actions based on updated pose estimates. By continuously updating the object pose through a combination of visual feedback and kinematic data from the robot arm, the proposed method can achieve a level of accuracy that is challenging for purely vision-based or learning-based approaches to match.

In summary, PCL with VFH and RGB-D data presents a balanced approach for this research project, offering a combination of detailed object analysis and real-time performance. Its ability to handle complex object shapes and dynamic scenes, when paired with push-based manipulations, positions it as a suitable choice for achieving the project's objective of enhancing robotic perception through refined 6D pose estimation. This method leverages the strengths of both traditional point cloud analysis and interactive manipulation, aiming to improve on what is possible in robotic vision systems.

Section 3. Experiment Method

3.1. Goal

The aim of this experiment is to;

1) Create an accurate pose estimation tool,
2) Calculate an approach vector to the target object, and
3) Identify and/or implement methods to improve pose estimation and push-based manipulation.

The target goal is to design a system that can accurately and reliably estimate an object's pose, calculate a target vector to connect with the object, and update its own internal pose estimation algorithm using external input to more accurately acquire future pose estimates.

3.2. The Test Environment

This experiment was conducted in an indoor laboratory under partially controlled conditions. It involved the following equipment;

1) A flat working surface approximately 1m x 1m,
2) A Universal Robotics UR5e Collaborative Robot (cobot) [Appendix.2],
3) Intel® RealSense™ Depth Camera D455 [Appendix 3],
4) Desktop PC; Ubuntu 20.04, AMD Ryzen 3600, Nvidia 2080S, B550M DS3H, 32GB RAM
5) A series of test objects; spray bottles, cubes, etc.
6) 3D-printed end effector attachment. Acts as a D455 camera mount and push manipulator.

The following figures are of the scene and test environment. The numbered labels correlate with the equipment listed above.

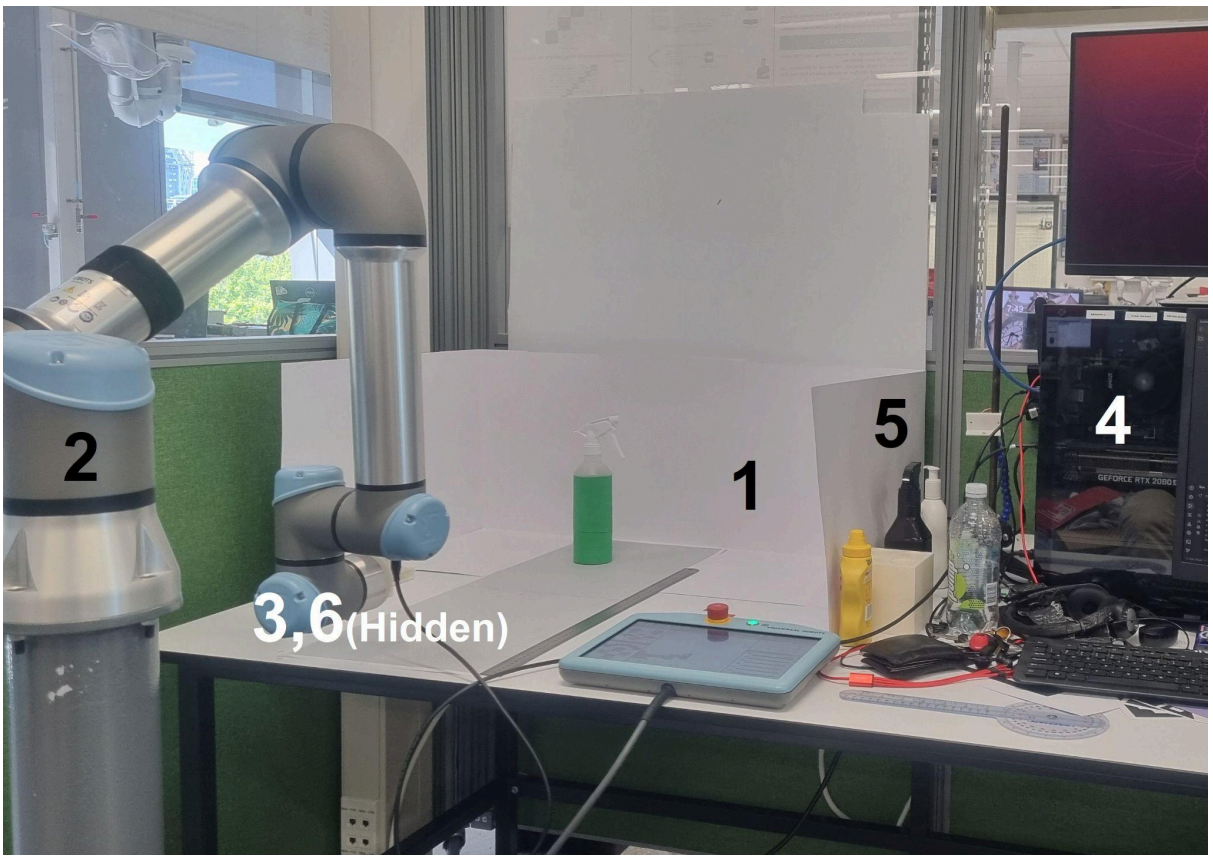
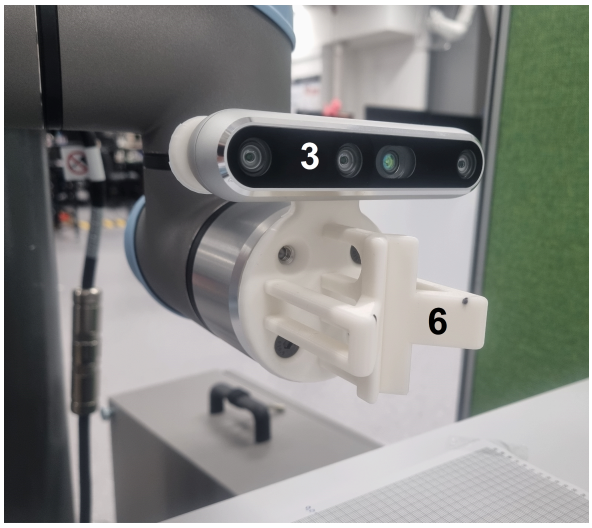


Figure 2. Labelled photograph of the laboratory testing scene and equipment.



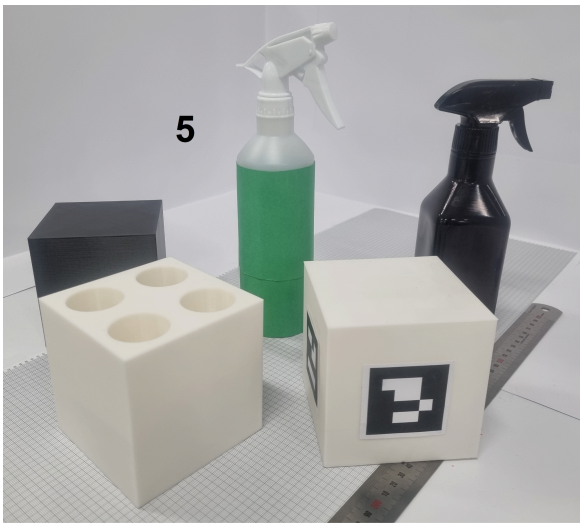


Figure 3. (Left) D455 Camera and End Effector attachment. Figure 4. (Right) Objects used during the experiment process.

The scene was enclosed in a box made from white printer paper and cardboard for support. This was done to reduce the amount of features and textures in the scene. Two A3 printouts of a scale accurate, 1mm grid were placed on the table in front of the camera, with a steel, 600mm ruler placed alongside the grid. The UR5e robot was mounted to a metal stand and

fixed to a position in front of the table. The end effector reference frame, known as the tool centre point (TCP) was placed above and parallel to the first line on the grid. This position was marked as 0,0 on the paper. It was programmed into the UR5e robot as the home position and is considered the world reference point (w.r.f) 0,0,0.

The joint position angles (in degrees) for the forward kinematics of the UR5e robot to get the TCP into this position were recorded as the following:

Base: -148.73	Shoulder: 33.78	Elbow: 122.02
Wrist 1: -87.77	Wrist 2: -279.46	Wrist 3: -0.72

Please note that the UR5e [Appendix2] does not specify the accuracy of these angles, by observations during testing however, it appears that the accuracy can be assumed to be within ±1 degree.

The pose of the TCP with respect to the base of the UR5e robot was recorded as being the following:

X: 316mm	Y: 369.36mm	Z: -93.41mm
RX: 3.816 rad	RY: -1.92 rad	RZ: 16.02

3.3. Designing the software

The software was designed and run on the operating system Ubuntu 20.04 - Focal. It imported libraries and dependencies from terminal commands such as apt-get, from zip-file installs, and repository pulls and builds.

Several functions across multiple files were created in c++ to perform different actions on the Intel Realsense depth camera stream and the point clouds extracted from it.

A high-level summary of some of the key functions are as follows:

1) Extract filtered clusters of point clouds from a scene
2) Analyse the VFH characteristic of the object point cloud
3) Save the point cloud and VFH of an object into a user-made object data library
4) Build a Kd-tree from a data library to use as a quick reference against new point clouds
5) Analyse the object point cloud to find an appropriate collision point for the TCP to connect with the target object

A GitHub repository was set up to manage the project and the full details of the code can be found at the following location: https://github.com/QuorZak/PCL-Pose[7].

3.4. Analysing the system behaviour

Once the test environment was set up, several objects were tested for their suitability and reliability to be detected by the depth camera. Notably, two spray bottles and two 100 mm cubes, each with a respective black and white variant (refer to Figure 3), were compared against each other. By viewing the depth stream, it could be observed that the black variants of each object were more reflective, and features less distinguishable under the sharp (less diffuse) lighting conditions. In the figures below, the two cubes were placed in the scene side-by-side, and at different orientations.

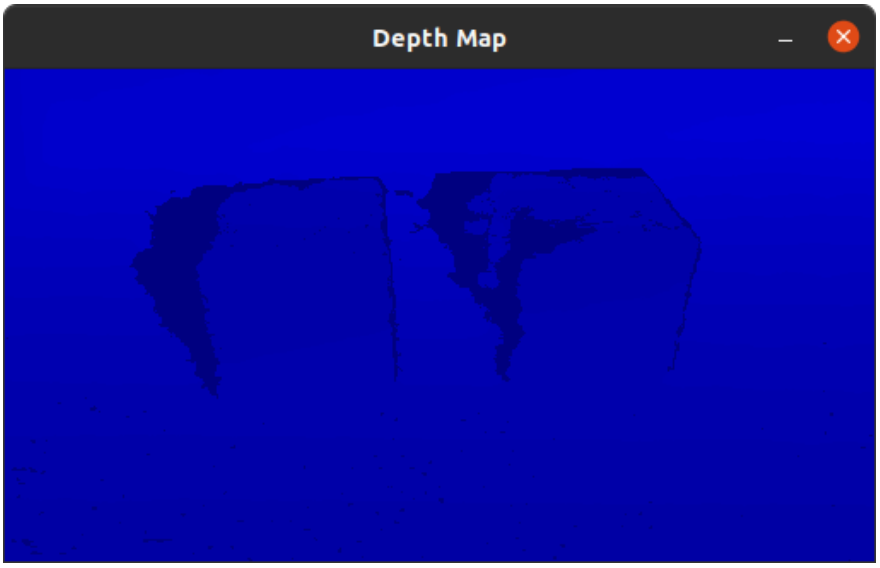


Figure 5. Depth stream of 2 cubes in scene. White cube (left) and black cube (right).

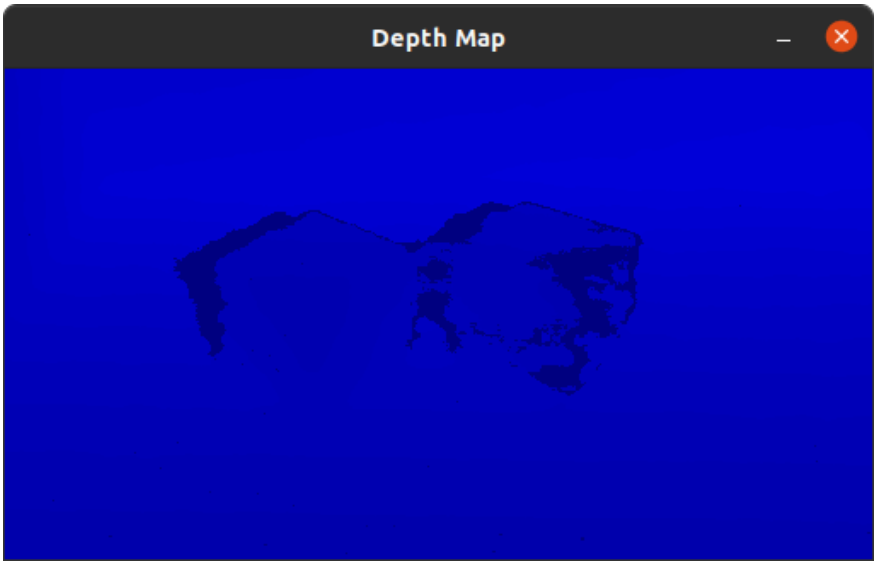


Figure 6. Depth stream of 2 cubes in rotated position. White cube (left) and black cube (right).

The white cube on the left was more matte than the reflective cube on the right, and was able to work well with the infrared light from the camera, as well as provide more consistent matching points for RBG data point pairing. Similar behaviour was observed in the black and white spray bottles.

Different viewing angles were also considered. At shallow angles (surface more perpendicular to the camera, there appeared to be more noise or a loss of depth information. In the side-by-side figure below, 2 cubes were recorded while the camera was perpendicular to, and approximately 100 mm above, the table surface, and then recorded at an approximately 45 degree and angle down from 300 mm above the table surface.

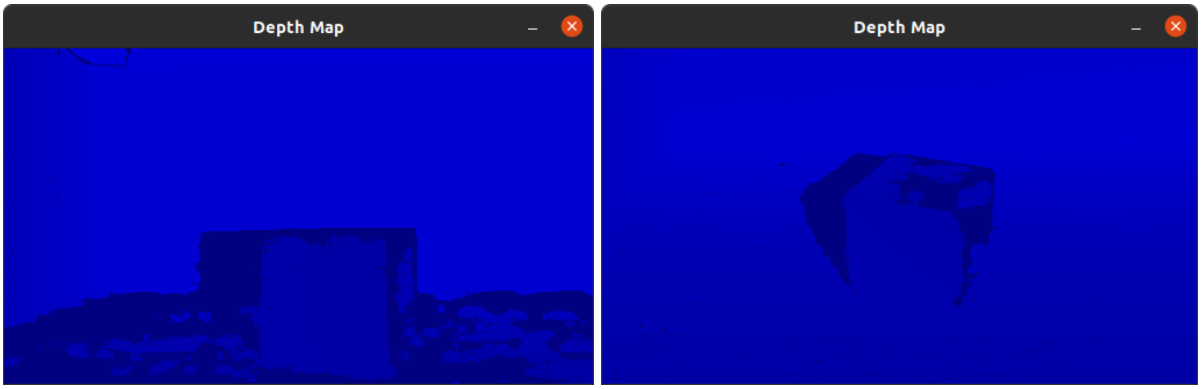


Figure 7. Depth stream of cubes in a scene viewed from perpendicular (left) and from 45 degrees downwards (right).

Note that in the left image in Figure 6, there is significantly more noise on the table's surface than when it is viewed from a steeper angle. This could indicate that the infrared component may become stretched 'thin' over a surface which has such a sharp viewing angle that it creates 'holes' in the surface.

Another thing observed during testing of the depth stream is the apparent 'shadow' behind each object in the scene. This was deducted to be because of how the Intel Realsense D455's lens and emitters are placed.

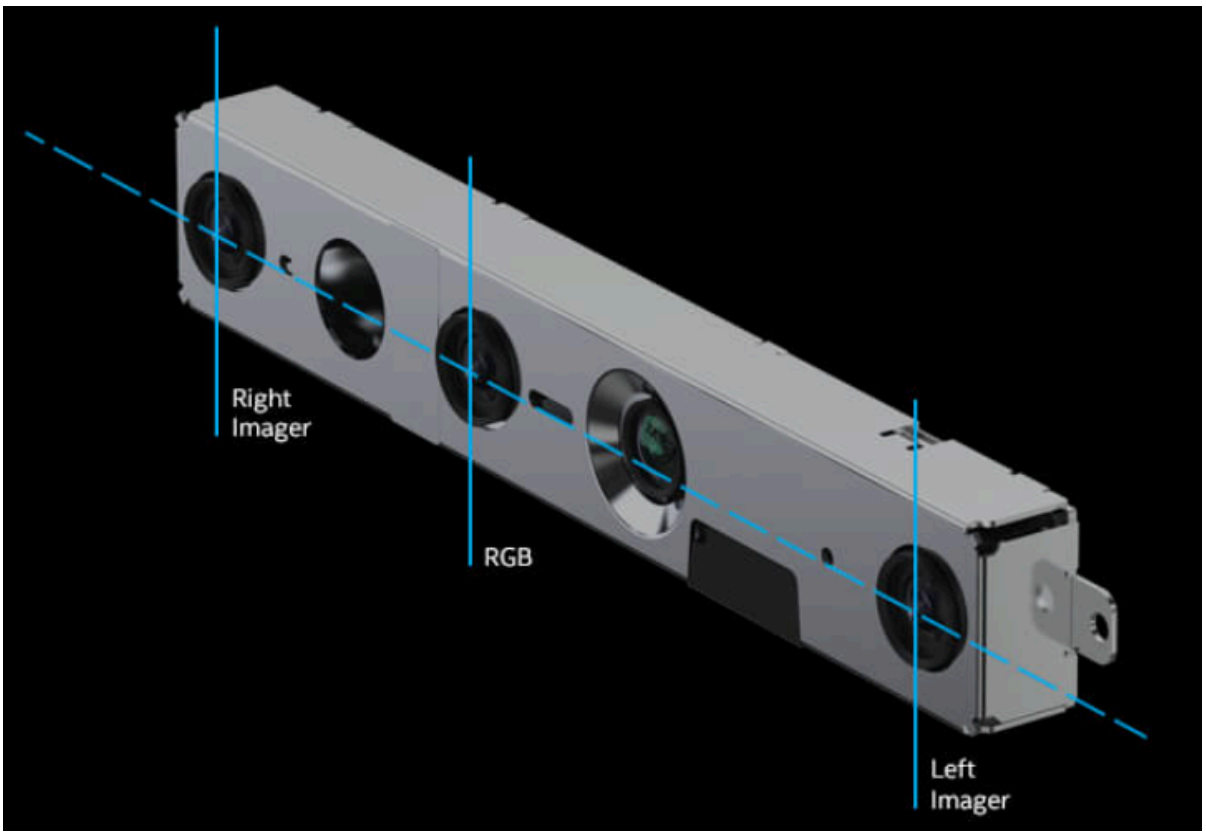


Figure 8. Intel Realsense D455 lens and emitter diagram[8].

Referring to the diagram in Figure 7 and referencing the behaviour of the camera in Figure 8, it was concluded that the left stereo imager (rightmost lens) was used as the base frame for depth imaging. Due to the offset between the left imager, the infrared emitter (2nd from the left in the above figure) and the right imager, there may be parts of the scene where no depth information can be gathered by the imagers. This may be due to the points not being seen by both imagers, and/or the infrared light not reaching the point.

Visualising the cylindrical spray bottle object in the scene through the Realsense viewer software program, provided by Intel and available to download, provided the following image (Figure 9). The object was placed directly in front of the camera and the camera was tilted slightly downwards. This image revealed that the Realsense camera used in the test was using the left imager as the base frame and centre-point for depth imagery. It also shows the coordinate system used by the camera, which is useful for understanding how to use and transform the depth data with respect to the robot and the world scene.

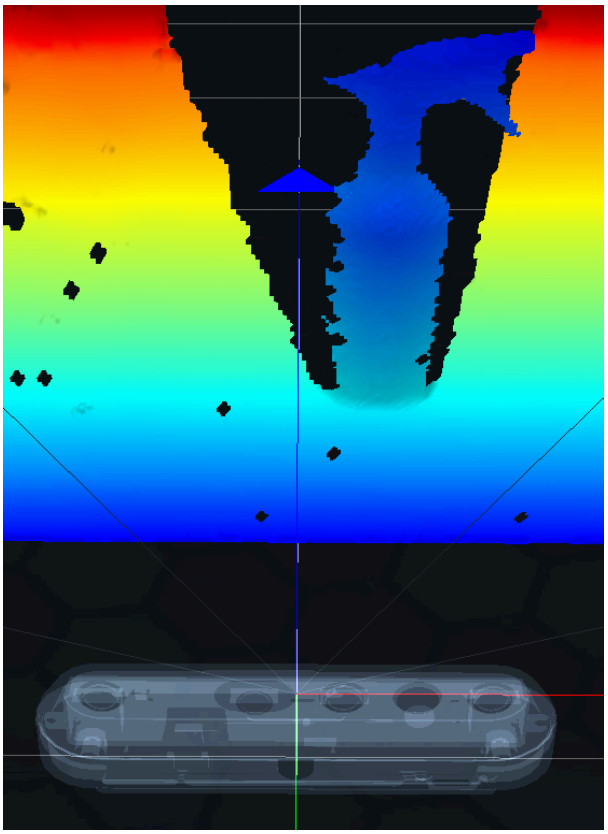
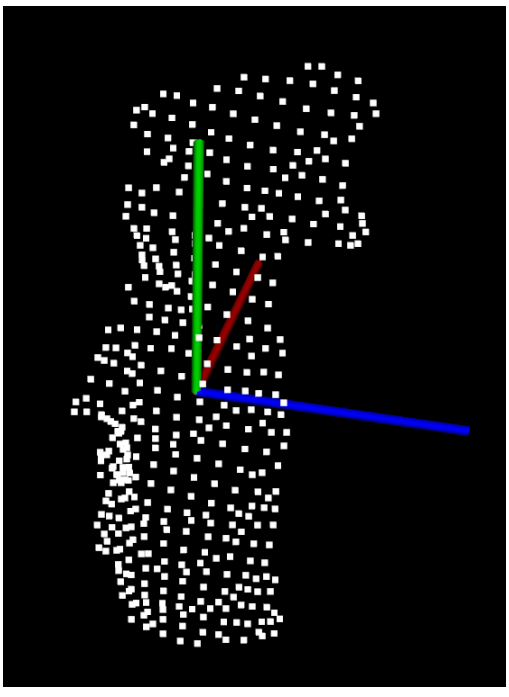


Figure 9. Intel Realsense Viewer depth image of spray bottle demonstrating offset and coordinate system of camera.

With a reasonable understanding of the depth stream, point cloud extraction was performed on the spray bottle. With some adjustments to the point cloud clustering parameters (such as leaf size and cluster tolerance) to remove noise and improve acquisition of the correct cluster, point clouds and their VFH, such as the following, could be extracted and saved in an object data library.

In order to estimate the pose, the coordinate system needed to be set for the stored point clouds in the data library. This was done by setting the object-facing angle in code which determined the front of the object, and by using the internal Intel Realsense accelerometer, the upwards direction could be ascertained (this is necessary because the camera was set to an angle of approximately 45 degrees). The aligned coordinate system was then placed at the centroid of the object and the point cloud was stored in the labelled data library folder.



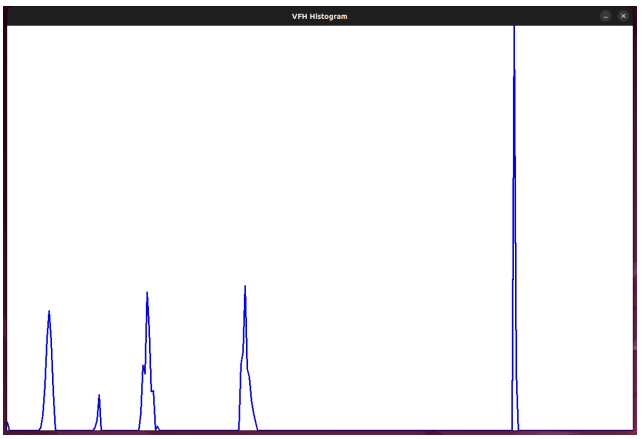


Figure 10. Spray Bottle VFH signature (Above).

Figure 11. Spray Bottle point cloud (Left).

Once enough point clouds to reliably recognise the target object in a scene were stored, adequate knowledge of the systems and reference frames was gathered, the functions described in 3.3 were implemented, offsets were set, translation matrices created, then testing for the systems pose estimation and approach vector could commence.

3.5. Testing method

For testing, the following procedure was conducted;

1) The object was placed in the scene at a known world location.
2) The TCP was returned to the home position above 0,0 as detailed in section 3.2.
3) A function was run to estimate the pose of the object and calculate a target point
4) Known object location and estimated object location was recorded

As testing progressed, refinements to the reference frame transforms, offset parameters and other variables were performed. This is because tests would reveal information which would lead to a better understanding of the system behaviour, and therefore potential improvements. The figure below shows the target object in the scene, at a known location and the TCP at its home 0,0 position.

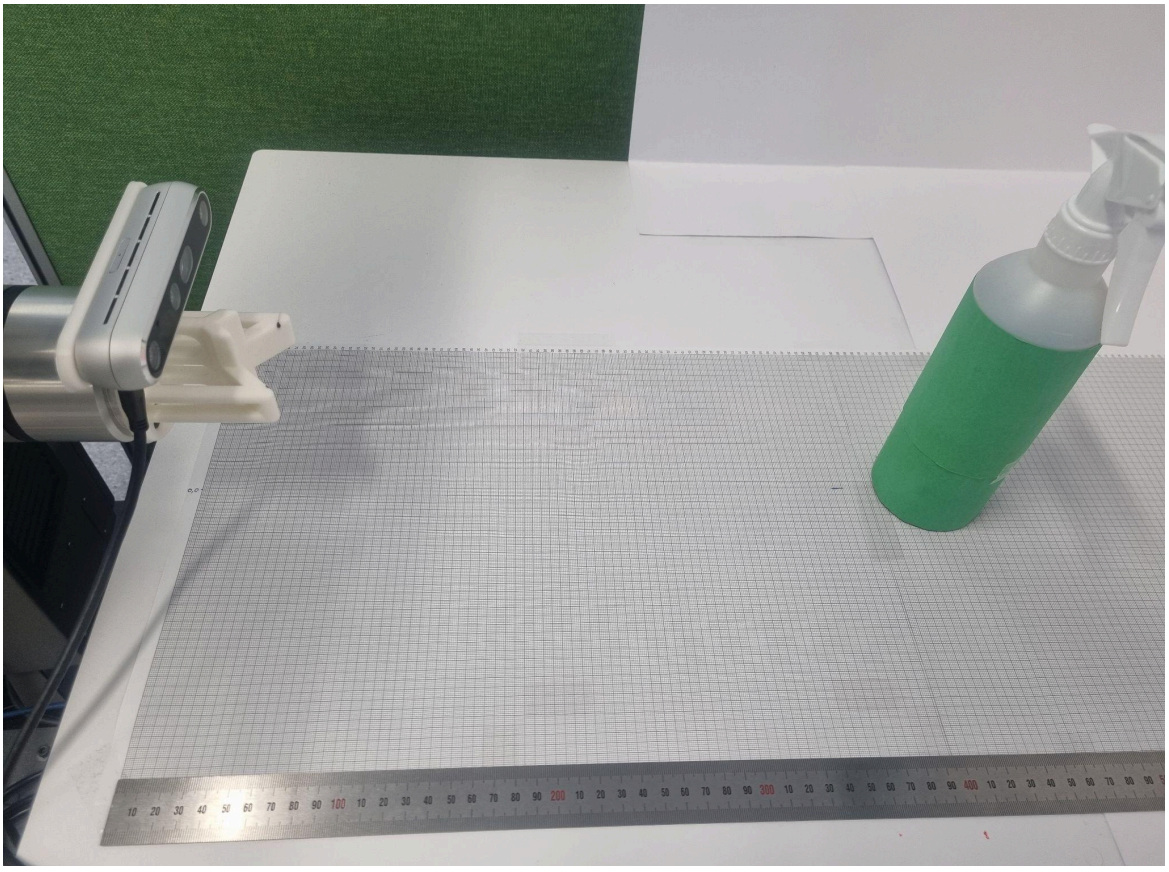


Figure 12. Photograph of scene with spray bottle object and TCP ready for testing.

Section 4. Results

4.1. Presentation of results

The full results of the testing procedure, including notes and records of each improvement or change to the system, are provided in Appendix 1.

In the final 4 test runs, which were performed using the most refined version of the pose estimation algorithm, the spray bottle object was placed in 4 different locations.

1) Centred, 400mm, 10 samples
2) Off-centre, 400mm, 10 samples
3) Off-centre, 600mm, 10 samples
4) Centred, 600mm, 10 samples

The diagram in the figure below shows the relative positions of the measured locations as well as the axes for reference.

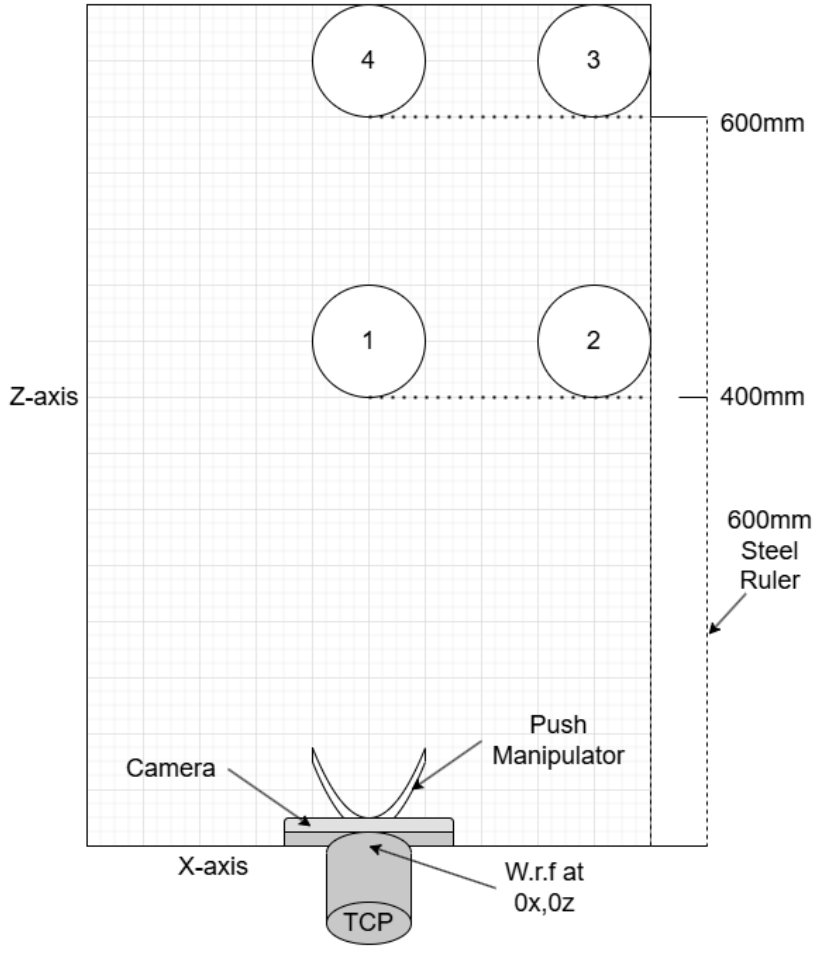


Figure 13. Diagram of the test scene, including 4 test positions, camera, TCP and w.r.f.

Results

The test results are:

Table 1. Centred, 400mm, 10 samples

Actual			Estimated		
x	y	z	x	y	z
0	0	400	2.48	-2.26	397
0	0	400	2.48	-2.26	397
0	0	400	-4.12	-20.98	398
0	0	400	-4.21	-17.29	398
0	0	400	4.89	-25.27	398
0	0	400	-3.98	-7.39	398
0	0	400	2.24	-10.18	398
0	0	400	2.31	-10.25	398
0	0	400	2.24	-22.34	398
0	0	400	4.89	-25.7	398

Table 2. Off-centre, 400mm, 10 samples

Actual			Estimated		
x	y	z	x	y	z
130	0	400	134.22	10.02	396
130	0	400	133.8	-2.04	396.24
130	0	400	134.22	10.02	396
130	0	400	144.73	-14.11	398
130	0	400	134.59	-22.16	399
130	0	400	134.43	-10.25	398
130	0	400	134.43	-10.25	398
130	0	400	127.78	-8.75	398
130	0	400	131.29	-22.41	398
130	0	400	127.85	-13.54	398

Results

Table 3. Off-centre, 600mm, 10 samples

Actual			Estimated		
x	y	z	x	y	z
130	0	600	134.7	10.44	599.67
130	0	600	138.58	10.39	599.33
130	0	600	138.69	-13.49	599.67
130	0	600	134.6	-9.53	599.33
130	0	600	138.62	-9.52	599.44
130	0	600	134.7	-5.52	599.67
130	0	600	130.82	-9.47	600
130	0	600	134.7	-9.5	599.67
130	0	600	138.47	-9.56	599
130	0	600	134.49	-9.56	599

Table 4. Centred, 600mm, 10 samples

Actual			Estimated		
x	y	z	x	y	z
0	0	600	-5.51	6.49	600
0	0	600	-10.1	10.6	601
0	0	600	-4.85	6.49	600
0	0	600	-8.84	14.47	600
0	0	600	-5.22	6.31	600.14
0	0	600	-5.01	6.34	600.12
0	0	600	-9.48	14.52	600.42
0	0	600	-4.91	14.37	599.22
0	0	600	-9.48	6.56	600.4
0	0	600	-8.84	10.48	600

The key metrics of the measurements (in mm) between actual world position and estimated position, for each test run, are as follows:

Average absolute diff.		
	x	z
1)	3.384	2.2
2)	4.508	2.46
3)	5.837	0.522
4)	7.224	0.286

Standard Deviation		
	x	z
1)	1.131	0.422
2)	3.745	1.045
3)	2.640	0.320
4)	2.274	0.359

4.2. Error

The UR5e cobot documentation reports its accuracy as Pose Repeatability per ISO 9283[9]. The repeatability is ±0.03mm[Appendix 2]. This error can be applied to both the home position of the robot as well as the actual position of the object. This is because the TCP end effector was used as an aid in setting the object in each test position.

The Intel Realsense D455 camera documentation reports a Depth RMS error(mm) of <2%[Appendix 3] and an accuracy (or absolute error) of ±2% at 4m[10]. It describes how these errors are affected by distance, lighting, depth image resolution, etc.[11]. While all factors of the environment were not recorded, the depth stream was configured to 848x480 resolution, which is reported to be optimal by Intel Developers[12]. The following graph shows how the RMSE for the D415 and D435 Intel Realsense cameras change with distance[11].

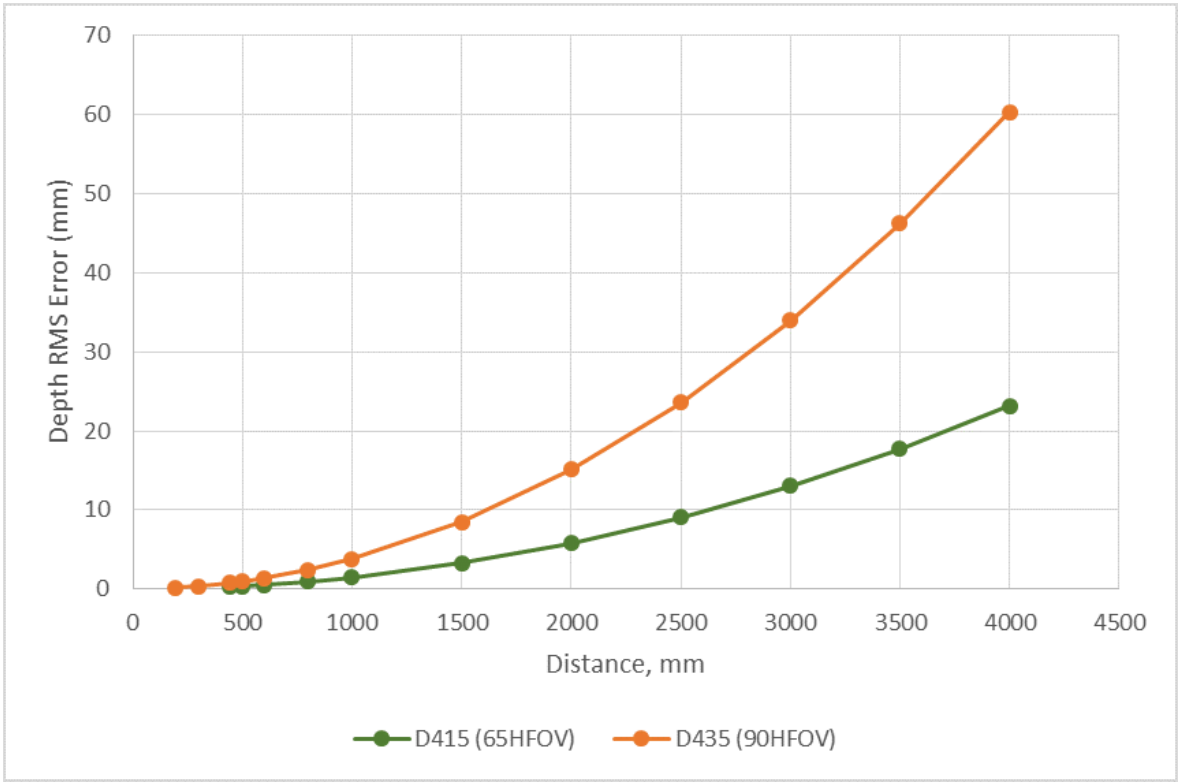


Figure 14. Graph of Intel Realsense D415 and D435 Depth RMSE over Distance.

The D455 is reported to be 2x more accurate than the D435 and scale linearly[12]. Taking into consideration this information, the documented 2% error at 4m, a conservative yet justifiable absolute error for the camera measurements can be estimated to be ±2mm.

Finally, the maximum resolution of the point cloud is determined by the leaf size. This is analogous to an averaging filter over a depth map to create distinct points that are spaced the distance apart of the leaf size. The final results were measured using a leaf size of 4mm. This has a most significant effect on the x and y axes as it determines the spacing of points when facing the object straight on.

Applying these error values gives the following results:

Average absolute diff.		
	x (±4.06mm)	z (±5.06mm)
1)	3.384	2.2
2)	4.508	2.46
3)	5.837	0.522
4)	7.224	0.286

Standard Deviation (incl. error 2.06mm)		
	x	z
1)	4.215	2.103
2)	5.582	2.310
3)	4.843	2.085
4)	4.653	2.091

Table 5. Average absolute difference between actual and estimated positions (Left) and Table 6. The standard deviation, including the error, across all estimations in a test location.

4.3. Discussion

Comparing the results between the actual pose of the object with respect to the TCP home w.r.f. and the pose estimated via the camera, results seem promising. The pose estimation results are within mm accuracy. All of the final readings on average appear to be <1 cm off from the actual value, with the exception of position 4 including maximum positive error. This means that the relationship between the robot's understanding of the object's pose and the camera's understanding of object pose (as it relates to push-based manipulation) is very close and shows that there is a fair possibility to use this method to allow the robotic system to gain useful information about the world outside of itself.

Observing patterns in the measurement data reveals that the readings are fairly consistent on a reading by reading basis.

With respect to the z-axis, the depth readings are very reliable, only ever off by 3-4 mm at most, and usually within 2 mm. This is great for arguably the most important axis with respect to push-based manipulation, as the x-axis can be made up for using a wider push

manipulator and the y-axis in this circumstance is only important when it comes to aiming for a point that is below the centre of gravity.

We can ignore the y-axis readings due to the statement made in the previous paragraph, however reviewing the data for assurance shows that the values remain within expected bounds.

The x-axis is the most variable and would be the area in need of most improvement. It fluctuates around 5 mm on average about the true centre of the object. However, this relatively low margin could be accurate enough to get the robotic arm close enough to the object to ultimately push or grasp it, depending on the end effector. The biggest factor in the x-axis inaccuracy is due to the leaf size, as mentioned in section 4.2. Attempts were made after the final 4 tests to lower the leaf size, but the right combination of point cloud cluster extraction parameters was not found such that the correct object in the scene could be reliably extracted. The lowering of the leaf size introduced too much noise and made the object indistinguishable from the scene. Attempts were made to use a lower leaf size with the final scene parameters, and also with an elevated camera with an angle of approximately 45 degrees downwards.

The x-axis was also affected the most by the interaction between the infrared sensor and the stereo imager sensor. Specifically the ‘shadow’ seen in Figures 4 through 6 which created noise on the left-hand side of the object in the scene. One test where the object was placed to the left (negative x direction) revealed a very high accuracy. Retroactively speaking, some more data from this location would have been insightful.

Further analysis of some of the patterns in the data reveal that often the nearest point calculated is not exactly the centre point as anticipated. For example, when the object is offset, the x-axis values seem to offset from the centre also. This is likely due to the perspective of the object with respect to the camera. In these circumstances it is important to see that the standard deviation is still quite low (around 4mm), so even though there may be a high average difference, the values do not vary too much. In the future the estimation algorithm could be updated to take into consideration this perspective change.

Section 5. Conclusions

5.1. Summary

The results provide compelling evidence to suggest that it is possible to use a stereo depth sensor, specifically the Intel Realsense D455 in conjunction with a Universal Robots UR5e cobot, for a number of pose estimation and push-based manipulation tasks. Specifically, this project supports the following capabilities:

- Target Object Identification: The system reliably identifies target objects within a scene using the Viewpoint Feature Histogram point cloud method.
- Object Pose and Orientation Understanding: The system can determine the pose and orientation of objects that have been added to an object data library.
- Collision Vector Calculation for Manipulation: The pose estimation algorithm enables the calculation of a collision vector that guides the robot to push or grasp objects in the scene.

Due to the relatively high accuracy of the object pose estimation, and this project's work to refine the relationship between the camera's estimation and the robot reference frame, testing has shown that the robot arm can work very well at receiving a collision vector from the pose estimation algorithm. The robot TCP is able to move into a position to push the object, typically within 5 mm of the object on the x-axis and 2mm on the z-axis. Additional work would need to be done in order to refine this further, including the addition of touch sensors, or other sensors that work well in close-proximity to the object.

Despite some of these successes, the system has limitations that affect its performance in certain conditions. The limitations of the system are:

- Object Identification Constraints: Identifying objects that are reflective, dark, or positioned in sharp lighting conditions remains challenging.
- Point Cloud Clustering Sensitivity: A practical minimum leaf size exists for point cloud clustering, below which noise makes object extraction highly challenging. Reducing the leaf size would require additional refinement of the cluster extraction algorithm to mitigate noise.

- Range and Perspective Limitations: The system's minimum detection range is approximately 355 mm from the camera's reference frame. Furthermore, camera angle significantly impacts performance, with shallower angles increasing noise on perpendicular surfaces. Additionally, there is a slight bias towards improved identification accuracy for objects located directly in front of or to the left of the left-hand stereo imager.

These findings highlight the system's robustness in controlled environments while also identifying areas where future enhancements could improve performance in more complex or dynamic scenes.

5.2. Future work

If this work were to be continued or incorporated into other works, the first step would be to download the repository from GitHub using the link provided in Reference 6[7]. The README.md file provides fair instruction on how to set up the code and gather the required dependencies. The README.md also details the different files and their functions.

With respect to improving the project as it currently rests, the following work is recommended to be performed;

Modify the code to:

- Average the target point estimation over X Samples to reduce variation
- Implement ROS wrapper to allow code to communicate with UR5e
- Continuously run a loop in either the main.cpp file or modify the manual_measure.cpp file to await the user's input and then send a command to the UR5e when prompted.
- Incorporate touch sensors on gripper/end effector to pass feedback information to code while arm is moving and collides with object. The existing pose manager class could be useful for managing the sensor-fusion and pose estimation updates.
- Refine the cluster extraction algorithm and adjust the parameters to make it more reliable.
- Improve the 6D pose estimation, which is currently using the centroid as the point to set the coordinate system, to set the coordinate system in a more robust way.

- Calculate a better approach vector which takes into consideration the 6D pose estimation in a more significant way.

Other actions would be to;

- Train a wider variety of objects and store them in an object data library.
- Incorporate a 2nd Intel Realsense camera running the same pose estimation algorithm and find a way to fuse the sensor information to potentially improve the accuracy of the object pose estimation.
- Perform tests on a cluttered scene with a variety of objects, at various angles, and under different lighting conditions.

References

[1] J. Guan, Y. Hao, Q. Wu, S. Li, and Y. Fang, "A Survey of 6DoF Object Pose Estimation Methods for Different Application Scenarios," *Sensors*, vol. 24, no. 4, pp. 1076–1076, Feb. 2024, doi: https://doi.org/10.3390/s24041076.

[2] Y. Lin, J. Tremblay, S. Tyree, P. A. Vela, and S. Birchfield, "Single-Stage Keypoint-Based Category-Level Object Pose Estimation from an RGB Image," *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, doi: https://doi.org/10.1109/icra46639.2022.9812299.

[3] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," *2011 IEEE International Conference on Robotics and Automation*, May 2011, doi: https://doi.org/10.1109/icra.2011.5980567.

[4] "Cluster Recognition and 6DOF Pose Estimation using VFH descriptors — Point Cloud Library 0.0 documentation," *Readthedocs.io*, 2024. https://pcl.readthedocs.io/projects/tutorials/en/master/vfh_recognition.html#vfh-recognition (accessed Oct. 21, 2024).

[5] "PCL (Point Cloud Library)," *Intel® RealSenseTM Developer Documentation*, 2024. https://dev.intelrealsense.com/docs/pcl-wrapper (accessed Nov. 08, 2024).

[6] F. Wang, C. Liang, C. Ru, and H. Cheng, "An Improved Point Cloud Descriptor for Vision Based Robotic Grasping System," *Sensors*, vol. 19, no. 10, p. 2225, May 2019, doi: https://doi.org/10.3390/s19102225.

[7] QuorZak, "GitHub - QuorZak/PCL-Pose: A project to investigate improving 6D pose estimation for push-based robots.," *GitHub*, 2024. https://github.com/QuorZak/PCL-Pose (accessed Nov. 08, 2024).

[8] "Introducing the Intel® RealSenseTM Depth Camera D455," *Intel® RealSenseTM Depth and Tracking Cameras*. https://www.intelrealsense.com/depth-camera-d455/

[9] *Manipulating industrial robots — Performance criteria and related test methods*, ISO 9283:1998, 1998. Available: https://www.iso.org/obp/ui/#iso:std:iso:9283:ed-2:v1:en

[10] "Intel RealSense D400 Series Product Family Datasheet," *Intel® RealSenseTM Developer Documentation*, 2024. https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet (accessed Nov. 10, 2024).

[11] "Tuning depth cameras for best performance," *Intel® RealSenseTM Developer Documentation*, 2024. https://dev.intelrealsense.com/docs/tuning-depth-cameras-for-best-performance?_ga=2.98684826.524766875.1731040155-786119102.1731040155#section-verify-performance-regularly-on-a-flat-wall-or-target (accessed Nov. 10, 2024).

[12] "Z-accuracy for D455," *Intel RealSense Help Center*, Dec. 07, 2021. https://support.intelrealsense.com/hc/en-us/community/posts/4412175528979-Z-accuracy-for-D455 (accessed Nov. 10, 2024).

Appendices

1. Measurement data collected from experiments

Measurements of Spray Bottle Object in Scene Detected by Intel D455 camera and UR5e robotic arm

Author: Zak Quor Date: 05/Nov/2024

Measurements in mm. World reference frame = w.r.f, x error = ±4.06mm, z error = ± 2.06mm

Actual/w.r.f			Camera			Abs. diff. between w.r.f and Camera			Standard Deviation	
Preliminary setup measurements										
x	y	z	x	y	z	x	z		x	z
22.73	0	312	14.97	7.99	306.3	7.76	5.7		5.431	2.333
14.97	0	306.3	14.89	0.17	303.9	0.08	2.4	Incl. error	6.780	3.113
					Avg.	3.920	4.05			
Update end effector offset to -60mm, adjust cam Z offset by -4.5mm										
33.68	0	295	20.07	25.88	295	13.61	0		1.155	0.314
20.07	0	295	33.89	41.2	295.32	13.82	0.32	Incl. error	4.221	2.084
33.89	0	295.32	19.96	17.46	295.21	13.93	0.11			
19.96	0	295.21	34.57	31.93	295	14.61	0.21			
34.57	0	295	49.75	24.6	295	15.18	0			

Appendices

Actual/w.r.f			Camera			Abs. diff. between w.r.f and Camera			Standard Deviation	
x	y	z	x	y	z	x	z		x	z
49.75	0	295	66.34	23.15	295.64	16.59	0.64			
66.34	0	295.64	81.59	29.89	295	15.25	0.64			
81.59	0	295	67.42	31.46	295.12	14.17	0.12			
67.42	0	295.12	80.61	31.85	295.82	13.19	0.7			
80.61	0	295.82	68.01	16.61	295	12.6	0.82			
						Avg.	14.295	0.356		

Decrease leaf size from 0.01 to 0.004, lowered cluster tolerance from 0.02 to 0.01, added denser model to library

17.29	0	308	16.68	26.11	305	0.61	3		4.716	1.517
16.68	0	305	16.95	9.72	302	0.27	3	Incl. error	6.223	2.558
16.95	0	302	14.1	4.35	298	2.85	4			
14.1	0	298	16.79	29.85	295	2.69	3			
16.79	0	295	4.97	22.21	295	11.82	0			
						Avg.	3.648	2.6		

Modify fixed world rotation RX from -89(deg) to -90(deg), reset object to same world location each time

0	0	295	17.04	25.81	295	17.04	0		3.570	0.000
0	0	295	13.68	46.83	295	13.68	0	Incl. error	5.406	2.060
0	0	295	-9.73	48.63	295	9.73	0			

Appendices

Actual/w.r.f			Camera			Abs. diff. between w.r.f and Camera			Standard Deviation	
x	y	z	x	y	z	x	z		x	z
0	0	295	17.36	26	295	17.36	0			
0	0	295	13.59	46.96	295	13.59	0			
0	0	295	13.82	46.68	295	13.82	0			
0	0	295	-4.72	46.01	295	4.72	0			
0	0	295	13.48	46.69	295	13.48	0			
0	0	295	-11.72	46.01	295	11.72	0			
0	0	295	13.48	46.69	295	13.48	0			
0	0	295	13.47	46.48	295	13.47	0			
0	0	295	17.8	26.1	295	17.8	0			
						Avg. 13.324	0			
Adjusted target area to be centre square of object, min depth filter of 0.3m affecting results - change to 0.15m										
0	0	300	1.01	-18.23	296	1.01	4		1.764	0.316
0	0	300	1.05	-17.58	295	1.05	5	Incl. error	4.427	2.084
0	0	300	1.35	-9.92	295	1.35	5			
0	0	300	4.91	-37.95	295	4.91	5			
0	0	300	4.91	-37.95	295	4.91	5			
0	0	300	4.91	-37.95	295	4.91	5			

Appendices

Actual/w.r.f			Camera			Abs. diff. between w.r.f and Camera			Standard Deviation	
x	y	z	x	y	z	x	z		x	z
0	0	300	1.39	-38.08	295	1.39	5			
0	0	300	1.55	-38.05	295	1.55	5			
0	0	300	1.28	-38.05	295	1.28	5			
0	0	300	1.28	-38.05	295	1.28	5			
					Avg.	2.364	4.9			
Increase X-axis placement location										
100	0	300	105.18	-21.85	297	5.18	3		2.309	0.247
100	0	300	105.18	-18.04	297	5.18	3	Incl. error	4.670	2.075
100	0	300	105.14	13.95	296.56	5.14	3.44			
100	0	300	100.97	1.76	296.97	0.97	3.03			
100	0	300	100.07	1.76	296.97	0.07	3.03			
100	0	300	105.12	13.94	296.52	5.12	3.48			
100	0	300	100.97	1.76	296.97	0.97	3.03			
100	0	300	100.98	2.01	296.74	0.98	3.26			
100	0	300	105.09	13.92	296.44	5.09	3.56			
100	0	300	100.97	1.95	296.44	0.97	3.56			
					Avg.	2.967	3.239			

Appendices

Actual/w.r.f			Camera			Abs. diff. between w.r.f and Camera			Standard Deviation	
x	y	z	x	y	z	x	z		x	z
Decrease X-axis placement location										
-100	0	300	-99.11	1.78	297.36	0.89	2.64		0.017	0.118
-100	0	300	-99.06	-18.1	297	0.94	3	Incl. error	4.060	2.063
-100	0	300	-99.05	-2.1	296.96	0.95	3.04			
-100	0	300	-99.06	-18.1	297	0.94	3			
-100	0	300	-99.06	-18.1	297	0.94	3			
-100	0	300	-99.06	-18.1	297	0.94	3			
-100	0	300	-99.06	-18.1	297	0.94	3			
-100	0	300	-99.05	-2.1	296.96	0.95	3.04			
-100	0	300	-99.06	-21.91	297	0.94	3			
-100	0	300	-99.06	-21.91	297	0.94	3			
						Avg.	0.937	2.972		

Discovered camera cutoff at 355mm regardless of filter setting, adjust baseline offset from 45mm to 47.5mm (is 95mm wide), improved detection of closest point to cam, adjust Z-axis offset for tool-push collision. Added dynamic world rotation RX calculation (accelerometer-based).

0	0	400	2.48	-2.26	397	2.48	3		1.131	0.422
0	0	400	2.48	-2.26	397	2.48	3	Incl. error	4.215	2.103

Appendices

Actual/w.r.f			Camera			Abs. diff. between w.r.f and Camera			Standard Deviation	
x	y	z	x	y	z	x	z		x	z
0	0	400	-4.12	-20.98	398	4.12	2			
0	0	400	-4.21	-17.29	398	4.21	2			
0	0	400	4.89	-25.27	398	4.89	2			
0	0	400	-3.98	-7.39	398	3.98	2			
0	0	400	2.24	-10.18	398	2.24	2			
0	0	400	2.31	-10.25	398	2.31	2			
0	0	400	2.24	-22.34	398	2.24	2			
0	0	400	4.89	-25.7	398	4.89	2			
						Avg. 3.384	2.2			
Increase X-axis placement location										
130	0	400	134.22	10.02	396	4.22	4		3.745	1.045
130	0	400	133.8	-2.04	396.24	3.8	3.76	Incl. error	5.523	2.310
130	0	400	134.22	10.02	396	4.22	4			
130	0	400	144.73	-14.11	398	14.73	2			
130	0	400	134.59	-22.16	399	4.59	1			
130	0	400	134.43	-10.25	398	4.43	2			
130	0	400	134.43	-10.25	398	4.43	2			

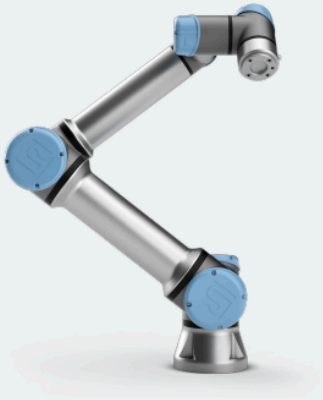
Appendices

Actual/w.r.f			Camera			Abs. diff. between w.r.f and Camera			Standard Deviation	
x	y	z	x	y	z	x	z		x	z
130	0	400	127.78	-8.75	398	2.22	2			
130	0	400	131.29	-22.41	398	1.29	2			
130	0	400	127.85	-13.54	398	2.15	2			
					Avg.	4.608	2.476			
Increase Z-axis placement location										
130	0	600	134.7	10.44	599.67	4.7	0.33		2.640	0.320
130	0	600	138.58	10.39	599.33	8.58	0.67	Incl. error	4.843	2.085
130	0	600	138.69	-13.49	599.67	8.69	0.33			
130	0	600	134.6	-9.53	599.33	4.6	0.67			
130	0	600	138.62	-9.52	599.44	8.62	0.56			
130	0	600	134.7	-5.52	599.67	4.7	0.33			
130	0	600	130.82	-9.47	600	0.82	0			
130	0	600	134.7	-9.5	599.67	4.7	0.33			
130	0	600	138.47	-9.56	599	8.47	1			
130	0	600	134.49	-9.56	599	4.49	1			
					Avg.	5.837	0.522			
Decrease X-axis placement location										

Appendices

Actual/w.r.f			Camera			Abs. diff. between w.r.f and Camera			Standard Deviation	
x	y	z	x	y	z	x	z		x	z
0	0	600	-5.51	6.49	600	5.51	0		2.274	0.359
0	0	600	-10.1	10.6	601	10.1	1	Incl. error	4.653	2.091
0	0	600	-4.85	6.49	600	4.85	0			
0	0	600	-8.84	14.47	600	8.84	0			
0	0	600	-5.22	6.31	600.14	5.22	0.14			
0	0	600	-5.01	6.34	600.12	5.01	0.12			
0	0	600	-9.48	14.52	600.42	9.48	0.42			
0	0	600	-4.91	14.37	599.22	4.91	0.78			
0	0	600	-9.48	6.56	600.4	9.48	0.4			
0	0	600	-8.84	10.48	600	8.84	0			
						Avg.	7.224	0.286		

2. Universal Robotics UR5e Cobot Datasheet



UR5e

Technical Specification

The UR5e brings ultimate flexibility to medium-duty applications with a payload of up to 5 kg and a reach of 850 mm.

Today, more than 90,000 UR collaborative industrial robots have been delivered to customers across industries and around the world. The e-Series of cobots brings incredible flexibility and unparalleled ease of use to your application.

Updated September 2024.

UR5e

Specification

Payload	5 kg (11 lbs)
Reach	850 mm (33.5 in)
Degrees of freedom	6 rotating joints
Programming	12 inch touchscreen with PolyScope graphical user interface
Power consumption (average)	
Maximum power	570 W
Moderate operating settings	200 W
Operating temperature range	Ambient temperature: 0-50°C (32-122°F)
Safety functions	17 configurable safety functions
In compliance with	EN ISO 13849-1 (PLd Category 3) and EN ISO 10218-1

Performance

Force sensing, tool flange/torque sensor	**Force, x-y-z**	**Torque, x-y-z**
Range	± 50.0 N	± 10.0 Nm
Precision	± 3.5 N	± 0.2 Nm
Accuracy	± 4.0 N	± 0.3 Nm

Movement

Typical TCP speed	1 m/s	
Pose Repeatability per ISO 9283	± 0.03 mm	
Axis movement	**Working range**	**Maximum speed**
Base	± 360°	± 180°/s
Shoulder	± 360°	± 180°/s
Elbow	± 360°	± 180°/s
Wrist 1	± 360°	± 180°/s
Wrist 2	± 360°	± 180°/s
Wrist 3	± 360°	± 180°/s

Features

IP classification	IP54
Cleanroom classification ISO 14644-1 *velocity and payload	Class 4 at 20%* Class 5 at ≥ 40%*
Noise	< 65 dB(A)
Robot mounting	Any orientation
I/O Ports	
Digital In	2
Digital Out	2
Analog In	2
Tool I/O power supply voltage	12/24 V
Tool I/O power supply	1.5 A (Dual pin) 1 A (Single pin)

Physical

Footprint	Ø 149 mm
Materials	Aluminum, plastic, steel
Tool Flange	EN ISO-9409-1-80-6-M8
Connector type	M8 8-pin female
Cable length (robot arm)	6 m (236 in)
Weight including cable	20,6 kg (45.4 lbs)
Humidity	≤ 90% RH (non-condensing)

Contact

Universal Robots A/S
Energivej 51
5260 Odense
Denmark

+45 89 93 89 89
sales@universal-robots.com
universal-robots.com

Teach Pendant

Features

IP classification	IP54
Humidity	≤ 90% RH (non-condensing)
Display resolution	1280 x 800 pixels

Physical

Materials	Plastic
Teach Pendant size (W x H x D)	300 mm x 231 mm x 50 mm (11.8 in x 9.1 in x 1.97 in)
Weight	1,8 kg (3.961 lbs) including 1 m of teach pendant cable
Cable length	4,5 m (177.17 in)

Control Box

Features

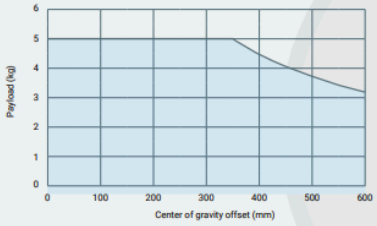
IP Classification	IP44
Operating temperature range*	Ambient temperature: 0-50°C (32-122°F)
Humidity	≤ 90% RH (non-condensing)
I/O Ports	
Digital In	16
Digital Out	16
Analog In	2
Analog Out	2
Quadrature Digital Inputs	4
I/O Power Supply	24V, 2A
Industrial Protocols	Modbus-TCP (Client/Sever) Ethernet/IP Adapter PROFINET device/PROFIsafe ROS/ROS2
Hardware Interfaces	Ethernet 1 Gb/s USB 2.0, USB 3.0 Mini DisplayPort Injection Moulding Machine Interface (SPI AN-146 & Euromap-67)
Power Source	100-240 VAC, 47-440 Hz

Physical

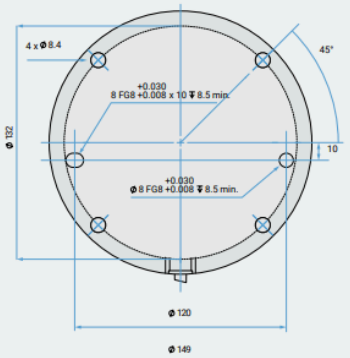
Control Box Size (W x H x D)	460 mm x 449 mm x 254 mm (18.2 in x 17.6 in x 10 in)
Weight	12 kg (26.5 lbs)
Materials	Powder coated steel

The control box is also available in an OEM version

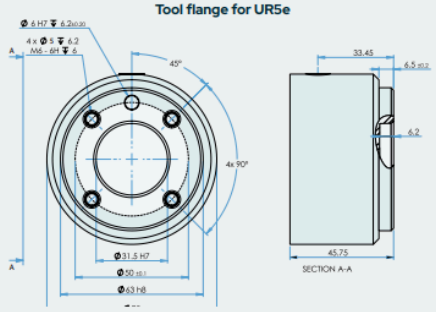
UR5e payload curve

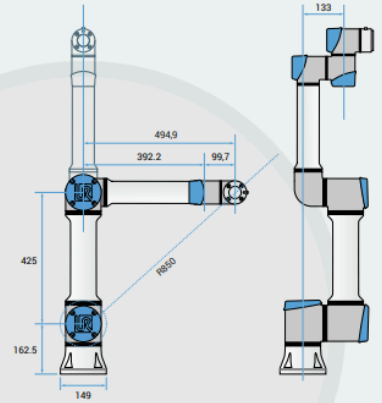


Interface for UR5e base flange



Tool flange for UR5e





3. Intel Realsense Depth Camera D455 Technical Specifications

Tech Specs

Datasheet

Features	**Use environment:** Indoor/Outdoor	**Ideal Range:** .6 m to 6 m
	Image sensor technology: Global Shutter	**Inertial measurement unit:** Bosch BMI055
Depth	**Depth technology:** Stereoscopic	**Depth Field of View (FOV):** 87° × 58°
	Minimum Depth Distance (Min-Z) at Max Resolution: ~52 cm	**Depth output resolution:** Up to 1280 × 720
	Depth Accuracy: <2% at 4 m[1]	**Depth frame rate:** Up to 90 fps
RGB	**RGB frame resolution:** Up to 1280 × 800	**RGB sensor FOV (H × V):** 90 × 65°
	RGB frame rate: 30 fps	**RGB sensor resolution:** 1 MP
	RGB sensor technology: Global Shutter	
Major Components	**Camera module:** Intel RealSense Module D450	**Vision Processor Board:** Intel RealSense Vision Processor D4
Physical	**Form factor:** Camera Peripheral	**Connectors:** USB-C* 3.1 Gen 1*
	Length × Depth × Height: 124 mm × 26 mm × 29 mm	**Mounting mechanism:** – One 1/4-20 UNC thread mounting point – Two M4 thread mounting points – Tripod