Coursework 1 Technical Report

Xinyue Hui

Student ID: k25123906

1 Introduction

This project presents a complete perception to manipulation pipeline for the UR5e robot arm. The system combines camera calibration, multi view Structure from Motion, point cloud processing, object localisation, and motion planning. The pipeline shows how visual information from ordinary RGB images can guide a real robot toward a successful grasp. The overall pipeline reflects key principles from the Sensing and Perception module, particularly in the areas of geometric reasoning, feature-based reconstruction, and perception-driven control.

2 Method

2.1 Camera Calibration and SfM Reconstruction

Camera calibration was carried out with a checkerboard pattern. The intrinsic matrix and distortion coefficients were recovered. Normalised points were obtained through

$$x_n = K^{-1} x_p. \tag{1}$$

Feature correspondences were detected with SIFT. The essential matrix was estimated to obtain relative camera motion. This follows the classical epipolar geometry formulation where the essential matrix encodes the relative rotation and translation up to scale. Triangulation produced a sparse 3D structure. Bundle adjustment refined all poses and 3D points. The system automatically loads the most recent reconstruction stored in the `results` directory, which ensures reproducibility.

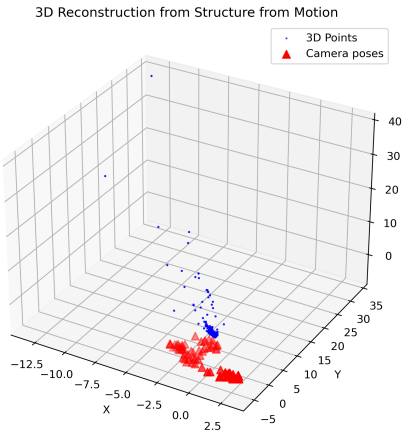

Figure 1: Sparse 3D reconstruction from SfM with camera poses.

2.2 Point Cloud Processing and Object Localisation

The raw SfM cloud usually contains noise. Therefore the geometric centroid is computed as

$$c = \frac{1}{N} \sum_{i=1}^{N} p_i. \tag{2}$$

Distant points are removed. The cloud extent is then measured. A uniform scaling step ensures that the largest dimension is close to 0.3 m so the cloud fits the UR5e workspace.

To align the cloud with the robot, a fixed translation is applied during publishing so the cloud appears in the `base_link` frame. The processed object location is defined as the centroid of the filtered cloud:

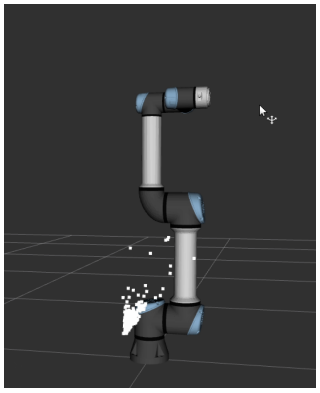

Figure 2: Processed point cloud aligned in the `base_link` frame.

$$c_{\text{obj}} = \frac{1}{M} \sum_{i=1}^{M} p_{\text{processed},i}. \tag{3}$$

These processed outputs form the geometric input for the subsequent motion planning stage.

3 UR5e Motion Planning

3.1 Joint-Space Representation and Pre-Grasp Pose

The UR5e robot configuration is represented by the joint vector

$$q = [q_1, q_2, q_3, q_4, q_5, q_6]^\top, \tag{4}$$

and any motion in joint space is described by a continuous trajectory

$$q(t), \qquad t \in [0, 1]. \tag{5}$$

MoveIt2 uses the OMPL RRTConnect planner to compute feasible paths under joint limits, velocity constraints, and collision avoidance.

Directly planning from arbitrary robot postures often leads to unstable behaviour or planner failure. To improve reliability, a manually verified and non-singular pre-grasp configuration is introduced:

$$q^{\text{pre}} = [-39^\circ, -98^\circ, 144^\circ, -87^\circ, 17^\circ, -93^\circ]^\top. \tag{6}$$

The first planning stage enforces

$$q(0) = q^{\text{start}}, \qquad q(1) = q^{\text{pre}}, \tag{7}$$

so that all subsequent motions begin from a stable, collision-free posture. This greatly improves the consistency of the RRTConnect planner and prevents the robot from entering configurations close to singularities or the table surface.

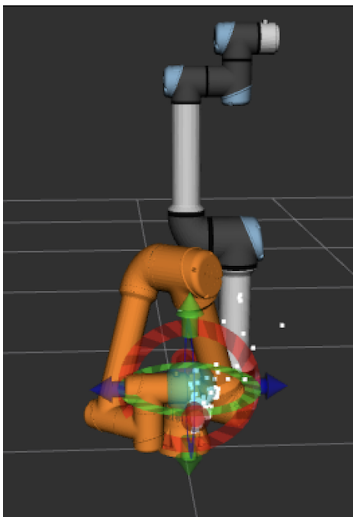

Figure 3: UR5e pre-grasp joint configuration used to stabilise motion planning.

With the robot placed in this verified pose, the system can reliably compute the later stages of the grasping sequence, including the IK-based approach towards the object and the final Cartesian descent.

3.2 Cartesian Approach and Grasping Strategy

To approach the object from above, a vertically offset Cartesian target is defined as

$$p_{\text{above}} = c_{\text{base}} + [0,\, 0,\, h]^{\top}, \tag{8}$$

where h is a predefined height offset.

MoveIt2 computes an IK solution for p_{above}, then generates a trajectory to reach the target pose with a consistent orientation. The robot then performs a downward Cartesian motion along the z-axis toward the object.

4 Technical Challenges

Several challenges appeared during development and were solved in sequence as the system was built. First the SfM reconstruction often produced clouds with inconsistent scale. Some clouds were large while others were small, and they did not match the UR5e workspace. A uniform scaling method based on the maximum cloud dimension solved this issue. Next the reconstructed cloud did not align with the robot coordinates. This caused errors in RViz and shifted grasp targets. A fixed translation in the publishing node ensured consistent alignment with the `base_link` frame.

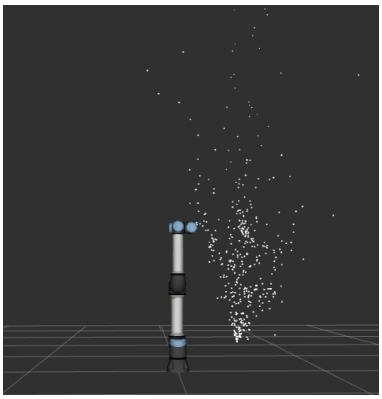

Figure 4: Misaligned point cloud before applying translation correction.

Planning stability also presented difficulties. RRTConnect frequently failed when the robot started from unstable postures. A manually verified pre grasp pose improved reliability. Furthermore the final approach used Cartesian motion, which reduced sampling failures. ROS2 related issues also appeared. Incorrect entry points, missing frames, and inconsistent resource paths caused nodes to fail. A careful restructuring of the package resolved these problems. This process highlighted the importance of consistent package structure and well-defined message interactions when building multi-module ROS2 systems.

5 Results and Future Work

The system successfully demonstrates the full pipeline: SfM reconstruction, point cloud filtering, object centroid estimation, and UR5e grasp execution. The processed cloud aligns correctly in RViz, and the robot performs a smooth, collision free sequence including pre grasp, approach, descent, placement, and return.

Although effective for single object scenes, several improvements remain possible. Future work may include point cloud clustering, geometric model fitting, depth completion, closed loop visual feedback, force sensing, and online trajectory refinement. The project therefore provides a strong foundation for more advanced manipulation systems.

Source Code Repository

All ROS2 packages and scripts used in this coursework are available in the following GitHub repository: `https://github.com/Hui0427/SAP_cw1_k25123906`

References

[1] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2022.

[2] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2004.

[3] Open Robotics, ROS2 Documentation. Available at: `https://docs.ros.org/`

[4] MoveIt Developers, MoveIt2 Documentation. Available at: `https://moveit.picknik.ai/`