3D Scanning with Fetch Robot

Qi Jiang
2022233182
jiangqi2022@shanghaitech.edu.cn

Xiao Han
2022233173
hanxiao2022@shanghaitech.edu.cn

Abstract

3D scanning technology, widely valuable in fields like quality control, reverse engineering, and cultural heritage preservation, is essential for creating precise digital models of physical objects. Traditional methods, however, often face limitations in resolution and capture range, leading to incomplete scans that miss the full shape of objects. This study introduces a novel automatic 3D scanning approach using an RGBD camera attached to a fetch robot's gripper, which leverages the robot's motion planning for detailed object scanning. The process involves analyzing point cloud normals to identify and rescan low-quality areas. The RGBD camera consistently maintains a 30 cm distance from the object, ensuring high-resolution results. Moreover, this method is adaptable for both object and scene scanning, enabling thorough and high-quality reconstructions.

1. Introduction

3D scanning is a technology used to capture the physical shape and appearance of an object or environment in three dimensions. The primary output of a 3D scan is a point cloud, which is a collection of data points in three-dimensional space. These points can be connected to form a mesh, a digital model that represents the surface of the scanned object and the mesh can be textured and colored if the scanner captures surface imagery. The application of 3D scanning spans various sectors. In manufacturing, it plays a crucial role in quality control and reverse engineering. The healthcare industry utilizes it for creating prosthetics and surgical planning. It's also instrumental in preserving cultural heritage, aiding in construction projects, and in the entertainment industry, where it facilitates the creation of digital models for films and video games.

Current 3D scanning methods commonly face resolution limitations tied to the distance of the scanning device from the target. For instance, scanning larger objects like a table or an entire room from a distance often leads to lower resolution results. This is a significant drawback, especially for interactive applications like virtual and augmented reality (VR/AR), video games, smart home technology, and embodied AI. In these fields, there is a strong preference for high-resolution scans that produce densely colored point clouds, enabling the reconstruction of high-quality meshes for the objects in question. This level of detail is crucial for creating immersive and accurate digital environments and experiences.

In our study, we employ a mobile fetch robot equipped with an RGBD Kinect sensor attached to its gripper for automated, high-resolution 3D scanning. The robot arm is designed to maneuver around the object at a close range, approximately 30 cm, to ensure high-resolution results. The process begins with the robotic arm conducting a long-range, coarse-grained scan. This initial scan is converted into an octomap, which aids in preventing arm collision during motion planning. For each scan, the octomap records the scanning distance and the normal direction of every point. Utilizing this data, we filter out low-quality points based on their normal direction and distance from the sensor, earmarking them for rescanning. In this rescanning phase, we prioritize the farthest points from the set for the robot arm to rescan. This step is repeated until a high-quality scan is achieved. The final stage involves merging all the raw scans to form a comprehensive, high-resolution point cloud, complete with color. This method significantly improves the quality and detail of the 3D scanning output, making it particularly suitable for applications that require precise and detailed digital representations.

2. ROS Dependencies

In this section, we introduce some ROS dependencies leveraged in our work.

2.1. Azure Kinect ROS Driver

The Azure Kinect ROS Driver represents a significant advancement in the realm of robotics and sensor integration, particularly emphasizing the Azure Kinect's depth sensing and RGB capture capabilities. Azure Kinect, a cutting-edge sensor developed by Microsoft, stands out for its exceptional depth sensing technology, which allows for precise three-dimensional environmental mapping and object

recognition. This technology is crucial for various applications, from navigating complex spaces in robotics to creating detailed 3D models of environments. In addition to depth sensing, Azure Kinect boasts a high-quality RGB camera, providing clear and accurate color imaging. This feature is essential for tasks that require color recognition or detailed visual analysis, such as object detection and tracking in dynamic environments.

Integrating these capabilities with the Robot Operating System (ROS) through the Azure Kinect ROS Driver opens up a new world of possibilities for robotics developers and researchers. The driver ensures that the intricate data captured by Azure Kinect's depth and RGB sensors are efficiently translated and made accessible within the ROS ecosystem. This integration enables advanced applications in robotic vision and spatial awareness, allowing robots to interact more effectively with their surroundings and perform complex tasks with greater accuracy. By combining Azure Kinect's sophisticated depth sensing and RGB capture with the flexibility and power of ROS, the Azure Kinect ROS Driver stands as a pivotal tool in advancing robotic capabilities and expanding the scope of what can be achieved in the field of robotics and automated systems.

2.2. Easy handeye calibration

The Easy Handeye Calibration package[4] is designed to simplify and streamline the process of calibrating the relationship between a robot's end-effector and a camera or any other sensor attached to it. Calibration is a critical step in robotic applications, particularly those involving precise manipulation or interaction with objects based on visual feedback. In traditional setups, this calibration process can be time-consuming and technically challenging, requiring significant expertise in both robotics and vision systems. The Easy Handeye Calibration package addresses this challenge by providing an intuitive, user-friendly interface and a set of robust algorithms that automate much of the calibration process. This package dramatically reduces the complexity and time required to achieve accurate hand-eye calibration, making it accessible to a broader range of users, including those with limited technical background in robotics or computer vision.

The package is designed to be highly versatile and compatible with a variety of robotic arms and vision systems, making it a valuable tool for a wide array of applications, from industrial automation to research and development in robotics. By leveraging advanced algorithms, the Easy Handeye Calibration package can accurately determine the transformation between the robot's coordinate system and the camera's viewpoint, enabling precise alignment and synchronization between the robot's movements and the sensor's data. This capability is crucial for tasks that rely on visual guidance, such as pick-and-place operations, assembly tasks, or any application where a robot interacts with its environment based on visual input. The Easy Handeye Calibration package not only enhances the efficiency and accuracy of these tasks but also opens up new possibilities in robotic applications by simplifying one of the most complex and critical aspects of robot setup and operation.

2.3. MoveIt

The MoveIt ROS package is a highly influential and widely used software in the field of robotics, serving as an essential tool for motion planning, manipulation, kinematics, and control. Developed within the Robot Operating System (ROS) framework, MoveIt is designed to facilitate complex robotic movement and interaction with the environment in a user-friendly and efficient manner. It stands out for its ability to handle both the computational and practical aspects of robotic movement, including collision detection, real-time planning, and manipulation. MoveIt's comprehensive suite of tools and libraries allows roboticists to develop sophisticated motion planning algorithms, manage robot kinematics, and integrate sensor information for reactive control. This makes it an invaluable asset in applications ranging from industrial automation to research in humanoid robotics, where precise and safe movement is paramount.

What sets MoveIt apart is its adaptability and ease of use, allowing it to be implemented with a wide variety of robots, from small desktop arms to full-size humanoid robots. Its modular architecture and integration with the ROS ecosystem enable developers to customize and extend its capabilities to suit specific application needs. MoveIt's interactive graphical interface and visualization tools further aid in simplifying the task of motion planning and execution, making it more accessible to users without deep expertise in robotics. This user-friendliness, combined with its powerful features, has made MoveIt a cornerstone in the ROS community, driving innovation and progress in the field of robotics. Whether for industrial applications, academic research, or hobbyist projects, MoveIt provides a comprehensive solution for robotic motion planning and execution, underscoring its significance in advancing the capabilities and accessibility of robotic systems.

2.4. OctoMap

The OctoMap[1–3] ROS package represents a pivotal development in the field of 3D mapping and environment modeling within the Robot Operating System (ROS) framework. At its core, OctoMap is a versatile and highly efficient library for generating and manipulating 3D occupancy grids. It excels in creating detailed, three-dimensional maps of environments by aggregating data from various sensors, such as LiDARs or RGB-D cameras. This capability is crucial in a wide range of robotics applications, including

autonomous navigation, obstacle avoidance, and complex scene understanding. The OctoMap package stands out for its unique approach to spatial representation, using an octree structure that allows for compact storage while maintaining high-resolution details in areas of interest. This efficiency in data representation and manipulation makes OctoMap an ideal choice for real-time applications where both accuracy and computational efficiency are essential.

Integrating OctoMap with ROS broadens its applicability and ease of use in robotic systems. Through this integration, roboticists and developers can seamlessly incorporate 3D mapping and environment modeling into their ROS-based applications. The package provides tools for updating and querying the map, enabling robots to interact with dynamic environments intelligently and adaptively. Furthermore, OctoMap's compatibility with other ROS packages, such as navigation and perception, allows for the creation of sophisticated and comprehensive robotic systems. The ability to efficiently process and utilize 3D spatial information opens up new possibilities in areas like autonomous exploration, search and rescue operations, and intricate manipulation tasks. Overall, the OctoMap ROS package is a testament to the advancements in robotic capabilities, offering a robust and scalable solution for 3D environment mapping and management in the ever-evolving field of robotics.

3. Implementation

3.1. System Description

In our approach, we utilize a Fetch robot arm equipped with an RGBD Kinect sensor mounted on its gripper. This setup enables the capture of dense point clouds with RGB color. As the Fetch robot arm moves, it facilitates the acquisition of multi-view, high-resolution scans of objects. These scans are then merged to form a comprehensive and dense point cloud, which can be utilized to reconstruct high-quality mesh of the objects.

3.2. Methodology Implementation Details

In this section, we introduce the detailed implementations of our project, including the device pre-processing such as sensor configuration and eye-on-hand calibration, and algorithm details.

3.2.1 Sensor Configuration

We equip the gripper of a fetch robot with an RGBD Kinect camera, allowing for automatic 3D scanning as it moves with the robot arm. To integrate an Azure Kinect Developer Kit into our existing ROS setup, we utilized the Azure Kinect ROS Driver package. This ROS node outputs various sensor data, from which we use the PointCloud2, optionally enhanced with color from the camera.

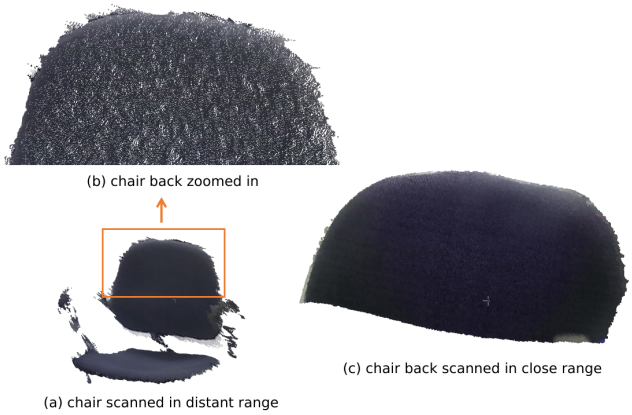
3.2.2 Eye-on-hand Calibration

Upon successfully obtaining sensor data from the Azure Kinect ROS Driver node, we perform the eye-on-hand calibration to compute the static transform between the reference frames of a robot's hand effector. With the sensor attached to the end-effector, we position a visual target, such as a calibration plate, at a fixed location. Initially, the apriltag ROS library is utilized for automatic detection of the visual target. Following this, the easy handeye package aids in computing the static transform matrix between the Kinect sensor and the gripper. This calibration matrix is computed by maneuvering the fetch robot arm, controlled via MoveIt, to capture the calibration plate from 25 distinct angles. Subsequently, we make the robot publish its own pose into tf.

3.2.3 Algorithm Details

After completing the preliminary setup of the device, we move on to the algorithmic aspect of our project. In this section, we will delve into the details of the algorithm implemented in our process.

- Move the Fetch robot arm to a fixed start pose and take a rough scan of the environment. We read the voxel cells' center points of the current octomap.
- Crop out the point cloud in the target region with a fixed 3D bounding box at the Region of Interest.
- Add the occupied cells into the PlanningSceneInterface to for robot arms' collision avoidance.
- Maintain a 3D voxel grid for the 3D region of interest. For each voxel, store the grid center position coordinates, its normal vector and its minimum scanned distance from its center point to the camera position.
- Divided all the occupied grids into two categories, grids that need to be scanned again and grids that are perfectly scanned. For octomap cells, the grids that need to be scanned again include the unknown cells, the frontier cells(a cell whose neighbours contain both unknown cells and occupied cells) and the grids that are scanned from a distant range. Since we temporarily failed to deserialize the octomap ros message, we simply take the grids scanned from a distant range as unfinished cells.
- For the unfinished cells, we select the furthest reachable cell as the next scan. We pose the camera pointing to the cells at the position of 20 centimeters along the normal vector. If the Inverse Kinematics fail to plan the motion, we mark the cell as unreachable and search for the next best scan.
- We take the scan and turn it into the base frame.
- Repeat step.3 to step.6 until complete high-resolution scanning.
- Take all the scans into a whole dense point clouds.



4. Experiment Results

4.1. Evaluation Metric

Since the motivation of our 3D scanning with fetch robot is to get a high-resolution dense scanned point cloud of an object, we choose the the level of hollowness as the evaluation metric.

4.2. Results

4.2.1 Metric Results

In the cropped target 3D region, the distant scan of the chair back is 5% hollow. Our scan in the close range is about 0.1% hollow.

4.2.2 Visualization

We merge the scans we get and crop out the target region. The obtained point cloud can be viewed as follow. (a) is the target chair scanned from the distant initial pose. (b) is the partial point cloud on the chair back of the scan (a). (c) shows the close scan taken automatically by the fetch robot at the position of 20 centimeters along the normal vector. The results shows that our method gains a much denser scan of the target object.

5. Conclusion

In this project, we concentrate on achieving high-resolution 3D scanning of objects using a fetch robot. Our method's limitation lies in not addressing unknown and frontier regions, primarily due to the lack of a Python interface for octomap data deserialization and our limited proficiency in C++. Consequently, we opted for voxelizing the point cloud rather than using octomap.

References

[1] Julie Stephany Berrio, Wei Zhou, James Ward, Stewart Worrall, and Eduardo Nebot. Octree map based on sparse point cloud and heuristic probability distribution for labeled images. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3174–3181. IEEE, 2018. 2

[2] Nathaniel Fairfield, George A. Kantor, and David S. Wettergreen. Real-time slam with octree evidence grids for exploration in underwater tunnels. *Journal of Field Robotics*, 24, 2007.

[3] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34:189–206, 2013. 2

[4] Roger Y Tsai, Reimar K Lenz, et al. A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration. *IEEE Transactions on robotics and automation*, 5(3): 345–358, 1989. 2