# ACTIVE ROBOT PERCEPTION FOR OBJECT POSE ESTIMATION

A Dissertation

by

JIE HU

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

| | |
|---|---|
| Chair of Committee, | Prabhakar R. Pagilla |
| Co-Chair of Committee, | Swaroop Darbha |
| Committee Members, | Won-jong Kim |
| | Xingyong Song |
| Head of Department, | Guillermo Aguilar |

August 2022

Major Subject: Mechanical Engineering

# ABSTRACT

When planning paths for robotic tasks involving interaction with an object, a key piece of information needed is the location of the object within the robot workspace. The process of obtaining the object location (both position and orientation) is referred to as *workpiece localization* in manufacturing, or more generally, *object pose estimation*. The object pose estimation process typically consists of two steps: data collection and pose estimation. Each step can be formulated and solved differently, or even separately, depending on the underlying process, the associated assumptions, and the work environment of the robot. In this work, an active robot perception framework that includes both data collection and pose estimation is proposed. The framework includes novel ways to (1) improve the accuracy of the estimated pose by collecting informative data and (2) plan subsequent sensor views automatically based on previously collected data. The data used in this work is in the format of point clouds. It is assumed that the 3D Computer-Aided Design (CAD) model of the target object is available. The object pose is estimated by registering the measured point clouds and the point clouds sampled from the CAD model.

The proposed active robot perception framework includes two main elements: view planning and pose estimation. View planning in this work includes generating and selecting sensor views and determining robot poses. Two sets of methods have been developed under the proposed framework. First, objects are assumed to have planar features, which are utilized for pose estimation. A plane-based point cloud registration method has been developed. Informative sensor view directions are defined based on the current pose estimation. Regions around the informative sensor view directions are discretized into voxels. Sensor view candidates are defined for voxels, and these candidates are further down-selected based on the kinematic feasibility of the robot to reach those views. A view gain is proposed to select the next-best-view from the view candidates. Simulations and experiments are conducted to evaluate the effectiveness of the pose estimation and view planning separately. The second set of methods is agnostic to the object geometries. Point cloud analysis in terms of quality and quantity is proposed to generate sensor view candidates.

The goal is to increase the estimated pose accuracy by improving the quality and quantity of the collected point clouds. Techniques from combinatorial optimization are utilized to determine the sensor views. Constrained nonlinear optimization is employed to calculate the robot poses corresponding to the sensor views. Experiments are conducted to evaluate the effectiveness of each component. The proposed methods are further compared with methods for reconstruction. The results from these comparisons reveal the differences between data collection for reconstruction and data collection for pose estimation. Generating sensor views based on the measured data is shown to have the following benefits: (1) view planning is less dependent on human experience; (2) sensor views can be generated efficiently and informatively for tasks with high variance; and (3) selecting sensor views offline to collect point clouds is avoided.

# DEDICATION

To my parents.

# ACKNOWLEDGMENTS

I would like to first thank my parents for their unconditional support over the years, not only for my education but in every other possible way. I would also like to thank my advisor, Dr. Pagilla, for giving me the chance to study and work in the lab. It takes a tremendous amount of time and resources to guide a Ph.D. student to complete their doctoral studies. You provided me with everything I needed to focus on my research. Dr. Darbha, it is an honor to have you as my co-chair. I have learned a lot about optimization from your class and our discussions. I enjoyed discussing problems with you. I would also like to thank Dr. Kim and Dr. Song for your support and invaluable feedback on my research. I cannot imagine my time here without my precious friends. Many thanks to y'all: Peng, Baik Jin, Maulik, Stephine, Mark, Cinthya, Crystal, Gema, Tiffany, Jay, Zongyao, Yalun, Orlando, and Angel.

CONTRIBUTORS AND FUNDING SOURCES

## Contributors

This dissertation was supported by a committee consisting of Professors Prabhakar Pagilla, Swaroop Darbha and Won-jong Kim of the Department of Mechanical Engineering, and Professor Xingyong Song of the Department of Engineering Technology and Industrial Distribution. All the work conducted for the dissertation was completed by the student independently.

## Funding Sources

Graduate study was supported by the Mechanical Engineering Department of Texas A&M University in forms of Teaching Assistantship, Fellowship, and by my advisor Dr. Pagilla in the form of Research Assistantship.

# NOMENCLATURE

| | |
|---|---|
| CAD | Computer-aided design |
| DOF | Degrees of Freedom |
| OLP | Offline programming |
| NBVs | Next-best-views |
| HMLV | High-mix, low-volume |
| HVLM | High-volume, low-mix |
| CMM | Coordinate Measuring Machine |
| SVD | Singular value decomposition |
| ICP | Iterative closest point |
| EGI | Extended Gaussian image |

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

# LIST OF TABLES

TABLE Page

# 1. INTRODUCTION

Robots have been widely used in manufacturing and automation applications, such as material handling, surface finishing, and welding, for their repeatability and flexibility in rapidly repurposing to different tasks. The research in robotics has accelerated significantly in the last several decades to develop more intelligent, robust, collaborative, and reliable robots. A typical robotic operation may include several of the following tasks: modeling, sensing, planning, optimization, and control. Modeling lays the mathematical foundations for calculations such as kinematics and dynamics. Sensing provides data for the robot to perceive and interact with the environment. Robot motions are planned to satisfy the task requirements and constraints of the system. Controllers are critical to achieving the planned robot motions.

Before planning motions for the robot to interact with an object, the location of the object within the robot workspace needs to be known. The process of obtaining the object location with respect to the robot is known as *workpiece localization* in manufacturing [4] and, more generally, *object pose estimation*. This dissertation uses workpiece localization when discussing manufacturing tasks and object pose estimation for general application contexts. The requirements on the object pose accuracy may vary for different applications such as robotic painting, grasping, and material removal. However, in all these applications, inaccurate object pose can cause quality and performance problems or even damage to the hardware, such as uneven colors during painting or non-uniform material removal.

Accurate knowledge of the object pose in the robot workspace is not only necessary but also beneficial. One such benefit is the reduced downtime by planning and programming trajectories offline. Robot trajectories can be generated via offline programming (OLP) when the object CAD model is available, and the location of the object is known. The positions of certain identifiable features on the CAD model, such as lines and surfaces, can be parameterized in the CAD frame, then transformed into the known robot (world) frame, based on which robot trajectories can be generated.

For the object pose estimation problem, one could pose two key questions: (1) what is the strategy to plan a view sequence to collect data? and (2) how to estimate the object pose using the collected data? Several methods exist in the literature to answer both questions. Planning a sequence of sensor views to collect data can be formulated as a *view planning* problem. Techniques have also been developed to estimate object pose using images or point clouds. There is a growing use of robots to automate processes as robots become more accessible. Both the tasks that robots are expected to complete and the environments that the robots work in are changing, usually characterized by environment variations and frequent changes in tasks. Those variations and changes may bring errors to the known object pose, which would further cause misalignment between the planned robot trajectories with the target geometries on the physical object if the workpiece location is not known accurately in advance. Thus, current view planning and pose estimation methods may not be efficient or even applicable to many robotics manufacturing and automation applications.

In the remainder of this chapter, we first discuss the challenges in object pose estimation and data collection when utilizing robots for different tasks where the object pose is not known a priori. Then, we discuss and review the existing relevant work. Finally, the scope and major contributions of the dissertation are provided.

## 1.1 Challenges in data collection and object pose estimation

There are known methods in both computer vision and robotics literature to address the data collection problem, such as methods from active perception, next-best-views (NBVs), and view planning. However, these methods have been designed mainly for 3D reconstruction and exploration of unknown spaces. Two key differences exist in how point clouds are collected in reconstruction and pose estimation. First, one collects an expansive set of point clouds to increase the coverage of the object to capture all the object features for reconstruction. Full coverage of the object is not necessary for pose estimation. Data collection can stop when the desired pose accuracy is reached. An example is shown in Fig. 1.1; partial point clouds of the Stanford bunny[1] can be

[1]Stanford bunny: `http://graphics.stanford.edu/data/3Dscanrep/`

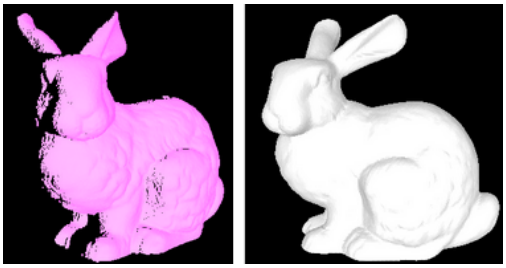


Figure 1.1: Point clouds of Stanford bunny. Partial point cloud can be used to estimate object pose and complete reconstruction of the object is usually not necessary.

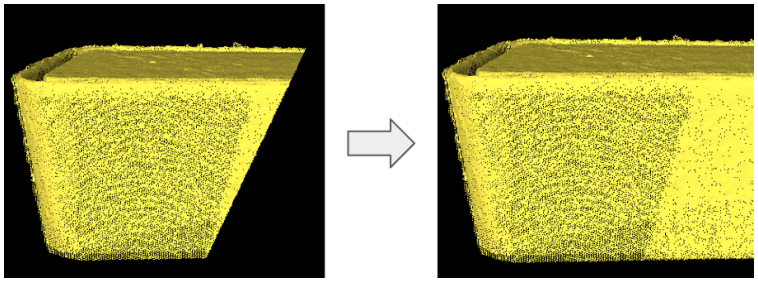


Figure 1.2: Point clouds of an organizer. Sensor views to increase volume or coverage and the collected point clouds may not help to increase the pose accuracy if the new point clouds do not exert additional constraints on the object.

used to estimate its pose and complete reconstruction may not be necessary. Second, current methods for reconstruction generate sensor view candidates either uniformly or randomly. A sequence of sensor views is planned by selecting views from these candidates. However, there are two potential issues when generating view candidates in such a manner. On the one hand, the generated view candidates may not be informative for pose estimation since volume or coverage increase does not necessarily increase the registration accuracy. An example is shown in Fig. 1.2; the new point clouds may not exert additional constraints on the object to improve the pose accuracy. On the other hand, the effectiveness of the previously generated view candidates may deteriorate when the object location is different from the location at which these view candidates were generated. Strategies to generate and optimize sensor views need to be developed for pose estimation when the object location is not known accurately.

There is also considerable existing work to solve the object pose estimation problem. The un-

derlying assumptions and complexity of the pose estimation problem may differ when working in different environments, such as structured vs. unstructured, dynamic vs. static, etc. In structured manufacturing environments, one could work with high-volume, low-mix (HVLM) or high-mix, low-volume (HMLV) workpieces. For HVLM workpieces, customized fixtures are usually utilized when mounting a workpiece in the robot workspace. Thus, the nominal workpiece location is usually assumed to be known a priori. Contact measurements can be employed to localize workpieces to the desired accuracy. However, for HMLV workpieces, custom fixtures are costly, and human-assisted data collection is usually required, which is time-consuming, tedious, and inefficient under potential object variations. In unstructured environments, the existing studies usually focus on the challenges in pose estimation due to clutters and occlusions [5]. Various methods using images or point clouds to estimate object poses have been developed [6] with a focus on utilizing point cloud registration for pose estimation. Assuming the point clouds are available, most current point cloud registration methods focus on developing robust algorithms to handle noisy point clouds. However, the registration accuracy can also be potentially improved by collecting point clouds that are less noisy and more informative. Further, the current estimate of the pose provides rich information to collect more data. Indeed, data collection is not stopped until the pose is estimated to an acceptable accuracy in practice. This dissertation describes some novel methods developed to address the aforementioned challenges in data collection and pose estimation.

## 1.2 Related work

This section reviews related work in the following areas: (1) workpiece localization and object pose estimation and (2) robot perception and view planning for data collection.

### 1.2.1 Workpiece localization

The workpiece localization problem may be defined as follows: a workpiece is placed in the work cell using fixtures, and a local coordinate frame is attached to the workpiece. Assuming an initial estimation of the workpiece location is known (nominal workpiece location), find the transformation from the robot frame to the workpiece frame such that the position and orientation

of the workpiece can be obtained. In this research, we assume the workpieces to be rigid bodies.

Depending on how the workpiece frame is defined, we can divide workpiece localization methods into two categories: direct measurement and feature matching. Direct measurement refers to the process of measuring the locations of some features on the workpiece or on the fixtures that are used to constrain or localize the workpiece and constructing a local coordinate frame by using the measured features. A priori knowledge of the workpiece is assumed, such as which planes forming the workpiece are orthogonal to each other, etc. Feature matching assumes that a CAD model of the workpiece is available. A nominal position of the workpiece is assumed to be known, and the CAD model is placed at the nominal location in the robot frame. Thus, the positions of the features on the CAD model are also known in the robot frame. After measuring some features on the workpiece that can be measured, a transformation can be calculated by matching the CAD model features with the measured features. Thus, feature positions on the physical workpiece are calculated by transforming the corresponding features on the CAD model by applying the obtained transformation. In this case, the workpiece frame is implicitly defined by the CAD model frame.

First, consider the widely used direct measurement approach in localizing workpieces when fixtures are used. If a workpiece has three orthogonal surfaces, then we can measure three points on one surface, two points on another surface that is orthogonal to the first surface, and another point on the surface that is orthogonal to both surfaces, which is referred to as the 3-2-1 principle [7]. If the coordinates of the measured six points are already in the machine/world frame, such as when using a Coordinate Measuring Machine (CMM) or CNC machine, then after constructing the workpiece coordinate frame with the intersection point of these three surfaces as the origin, the transformation between the workpiece frame and the robot frame is obtained. However, if the measured data is in the sensor frame, another transformation from the sensor frame to the robot frame is required. The 3-2-1 principle is still used in machining to localize workpieces because of its simplicity in localizing workpieces with orthogonal surfaces. Six physical locators corresponding to six measuring points are needed to constrain the workpiece. Other variations, such as 4-1-1, 4-2-1, and N-2-1, are also used for workpieces with different geometries, and dimen-

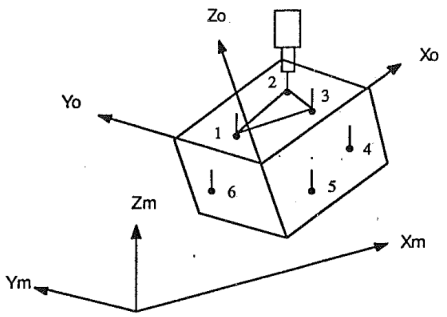



Figure 1.3: 3-2-1 convention method of finding a coordinate transformation [1]

sions [7]. Fixtures can be customized to meet the machining requirements; a detailed review of the state-of-the-art fixtures in manufacturing is provided in [7]. The 3-2-1 principle is sensitive to surface conditions and measurement errors due to the use of orthogonal surfaces. The localization accuracy predominantly depends on the machined locators and measuring errors, and to a large extent, it can be considered a fixture design problem.

In this dissertation, we consider workpiece localization by employing the feature matching approach, which has some similarities to the registration problem in computer vision. There are several differences between the workpiece localization problem and the registration problem. As considered in the computer vision literature, registration is typically treated as a part of the recognition problem – identifying the type of the object from a given set of objects and finding its position and orientation. While registration is more of a "by-product" of the recognition process, workpiece localization has to be considered by itself in robotic manufacturing [8]. Besides, in recognition, the initial estimation of the workpiece location is not assumed to be known while it is a reasonable assumption for the workpiece localization problem in manufacturing [1, 9]. Thus, global minimization is required for registration in most cases, while local minimization is adequate for workpiece localization [10]; the correspondence between measured data and CAD model features needs to be identified first in recognition while this relationship is assumed to be known in workpiece localization [11, 12, 8, 13, 14, 1]. Due to these assumptions in computer vision, some early work focused on approximating 3D objects using planes or polyhedra and using search methods

such as tree search and clustering to identify the correspondence between the approximated planes to recognize and locate 3D objects [15, 16, 14, 17, 11, 12, 18].

The workpiece localization problem has been mostly formulated as a *least squares problem*: A set of features $P = \{\mathbf{p}_i\}, i = 1, \ldots, n$ on the physical workpiece are measured and known in the robot frame $C_w$. Given the corresponding features $Q = \{\mathbf{q}_i\}, i = 1, \ldots, n$ on the CAD model with known position/orientation in the local coordinate frame $C_m$ and the transformation $\mathbf{T}_w^m$ between $C_m$ and $C_w$ is known, find the rigid body transformation matrix that minimizes the sum of the distances between the measured features $P$ to their correspondent features in $Q$. The correspondent features, for example, could be low-level features such as points, lines, edges, boundaries, or high-level features such as curvatures, straight dihedrals, etc. [8]. A variety of methods based on Singular Value Decomposition (SVD) can be found in [19, 20, 21, 22, 23, 24, 25, 26]. Besides, a near-optimal probing strategy that can determine the best set of points from a given points pool was provided in [9]. Similarly, a near-optimal/optimal probing strategy was proposed in [27, 28] to decide on probing locations and the number of points to measure since both affect the localization accuracy. Recently, a localization strategy that combines active sensing and contact-based workpiece localization was described in [29]; a probing strategy is selected based on the expected information gain associated with that strategy, and automatic localization of the workpiece is obtained by updating the belief of the transformation between the local coordinate frame assigned to the workpiece and the machine/fixed frame.

### 1.2.2 Object pose estimation using point clouds

The output of object pose estimation is a transformation matrix that describes the translation and orientation of the object coordinate system with respect to a reference coordinate system. In computer vision, images and point clouds have been used for object pose estimation. Point clouds are used in this work due to the wide availability of vision sensors that generate point clouds. In the following, we review existing pose estimation methods using point cloud registration.

Similar to the workpiece localization problem, the calculation of the transformation between two point clouds is straightforward when the correspondences between the two sets of points are

known.

One well-known method is the Iterative Closest Point (ICP) [30] and its variants [31, 32, 33]. The ICP algorithm iterates between finding the point correspondences and calculating the transformation using the correspondences. The obtained transformations from the ICP algorithm are usually local minima. Generalized-ICP is proposed in [32] to model locally planar surfaces in the scanned point cloud and the model point cloud, and this approach is reported to be more robust to incorrect correspondences. An ICP variant based on sparsity inducing norms is proposed in [31] to reduce the sensitivity to outliers and missing data. Another variant based on uniform sampling of normal vectors is proposed for the registration of nearly-flat meshes with small features in [33]. Other ICP variants are also classified and evaluated in [33].

One attempt to find the global transformation is by utilizing optimization techniques. One improvement of the ICP is the Go-ICP [34] which uses branch-and-bound techniques to find the global optimum. However, the computation time is significantly higher when compared to other ICP variants. Mixed-integer programming has also been used to find the correspondences in point cloud registration by relaxing the rotation matrix using McCormick relaxation [35] and convexification of rotation matrix constraints [36]. Convex semidefinite relaxation is utilized to reformulate the registration problem in [37], which returns global solutions when matching two isometric shapes.

Extensive efforts have been devoted to develop methods for finding the correspondences between two sets of points. One branch of methods addresses the problem by defining descriptors for each point and a metric to quantify the closeness between the descriptors. Points that have the closest descriptor values are considered as correspondences. The most well-known descriptors include: Spin images [38], which represent oriented points with 2D images that contain local surface patches, Fast point feature histograms (FPFH) [39], which uses histograms to describe the local geometries, Signature of Histograms of OrienTations (SHOT) [40], a four-dimensional feature that parametrizes geometrical relation of an oriented surface-point pair is proposed in [41]. Oriented point pairs are utilized to form a global model description in [42]. Learning based methods have

also been developed to learn descriptors that can be utilized for point cloud registration. A descriptor is learned by using a local volumetric patch in [43]. A deep learning network is proposed that leverages the alignment and attention mechanisms to learn feature correspondences from GPS and inertial navigation system (GPS/INS)is proposed in [44]. A voxelized smoothed density value (SDV) representation is utilized in the convolutional layers to learn 3D rotation invariant descriptor in [45]. Some other learning based methods have also been proposed to register point clouds. A 3D fully-convolutional network is used to compute the proposed fully-convolutional geometric features in [46] to find geometric correspondences. The Deep closest point (DCP) approach is proposed in [47], which embeds point clouds into a high-dimensional space and using attention module to encode contextual information. In [48], the Lucas & Kanade (LK) algorithm is modified to be used with PointNet [49] to learn features for point cloud registration. A method called PoseCNN is proposed in [5] to estimate the decoupled 3D translation and rotation based on semantic labeling. A review of pose estimation methods using point clouds can be found in [6].

### 1.2.3 Robot perception and planning

In workpiece localization, the number and locations of the measured points on the workpiece affect the localization accuracy in the workpiece localization problem; a planning strategy for taking measurements is thus necessary [27, 28, 50]. As proposed in [27], both locations of probing and the number of probing points affect the reliability and performance of the workpiece localization algorithm; based on the localization accuracy analysis, a maximum determinant planning strategy was proposed and a reliability analysis was conducted to determine the sufficient sample size needed to reduce the uncertainty of the localization error to a given value. A sampling point planning method was proposed in [28] to minimize the influence of workpiece surface errors on the localization accuracy. Six frame-invariant norms were proposed to quantify localization accuracy. Then, two criteria were designed to reflect the sensitivity of the localization accuracy to sampling errors at measuring points. Given a point set, the sample point planning problem was formulated as a *combinatorial problem* (select a subset of points from a given set), and solved by a floating forward searching algorithm. It was also observed that due to data saturation effect, if the number

of sampling points is large enough, the effect of optimization will be insignificant.

Most of the workpiece localization research has assumed a set of measured points is already available, and planning is based on that given point set. This approach has many issues: it is time-consuming to obtain good quality data in practice, the given data points may not be adequate to describe the object, and may require an experienced human operator to collect this existing data. In this dissertation, we do not assume data are available a priori and use tools in active perception and view planning to dynamically determine measuring locations to collect the data needed. There are several benefits of this approach, including (1) only measuring the required amount of data for localization, avoiding computation of redundant data; (2) automatic data collection using active perception negates the need for an expert human operator for data collection. In order to do so, relations between localization accuracy and sensor views need to be established, as well as the criteria used to evaluate the sensor views. We briefly review the related developments and technical tools behind active perception and view planning in the literature.

The concept of *Active perception* was originally defined in [51] as "an intelligent data acquisition process", and "an agent is an active perceiver if it knows why it wishes to sense, and then chooses what to perceive, and determines how, when and where to achieve that perception" [52]. Active perception includes five components related to data acquisition: why, what, how, when, and where. Thus, "An actively perceiving agent is one which dynamically determines the why of its behavior and then controls at least one of the what, how, where and when for each behavior" [52]. Each of the five components corresponds to a set of problems to be solved, and a significant number of methods are available in the existing literature to address them. For example, active control of the actuators or other parts of the robotic system, active control of the sensors and sensor parameters for the current tasks, active selection of sensor poses that are most appropriate for the current tasks, etc. Readers are referred to [52] for a detailed discussion of the five components and a review of active perception. The main inspiration from active perception is to use the information of the collected data as feedback to determine the robot motion for further data collection.

View planning studies consider how to decide the sensor views for a given task while consider-

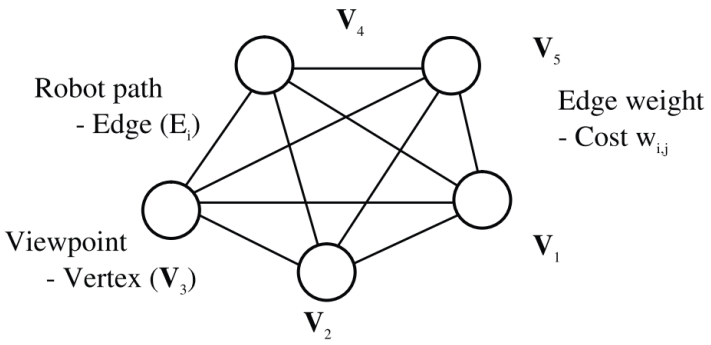



Figure 1.4: Sensor planning using a graph [2].

ing the associated requirements and constraints. Many view planning studies have focused on 3D modeling, reconstruction, inspection, and exploration [53, 54, 55, 56, 57, 58, 59, 60, 61]. The two main components of view planning are generating the candidate views and evaluating those views. The candidate views can be sampled randomly [55], uniformly on a hemisphere around the point of interest [56], or using frontiers (the boundary between known free space and unknown space) [62, 61]. The criteria for evaluating views include (1) view overlap, and shadow effects [53]; (2) occlusion, percentage of overlap for registration, amount of new surface (using ray tracing), and the travel distance from the current sensor location [55]; entropy reduction or information gain [58, 57]; and unmapped volumes or uninspected visible surfaces [59]. Examples of using voxel maps generated based on available CAD models to plan the scanning process in inspections can be found in [63, 64]. Incident angle has been used in view planning for reconstruction to evaluate voxel quality [65] and determine scanning direction [66]. Vertices that have large incident angles are removed in [67]. In stead of evaluating meshes or voxels, incident angle is used to predict if points are outliers using a dynamic angle interval in this work.

The *constraints* used in the view planning problem can result from the limits on sensors, robots, workpiece, and environment (work cell in manufacturing). Due to hardware limitations, factors such as resolution, field-of-view, view angle, field-of-depth, and light source need to be considered when using vision sensors [68, 69]. For example, the sensor field-of-view could constrain the

entire object to be not contained within the view of the sensor, the view angle limits the maximum angle formed between the surface normal and sensor beam. When sensors are mounted on a robot, kinematics and reachability should be considered. Occlusion of the workpiece due to other objects within the field of view could be another issue to consider.

Other related work includes approaches that rely on visibility maps, modeling the sensing process as a stochastic process, using graph theoretic techniques. The concept of a visibility map was defined in [70] by considering the geometry of the tool, which helps in reducing the setup times by exploiting the potential overlap between visibility maps. In [71], estimation of the 3D object position was formulated as statistical estimation of the geometric primitives (line, sphere, plane) that were used to model the target object. Probability distribution functions of the measured geometric primitives were used to formulate the noise mechanisms in measurements and modeling to calculate the maximum likelihood estimates of the parameters for geometric primitives. Bayes rule is also used to estimate the position and orientation of an object in a model-based active perception and sensing planning in [72]. A graph-based merging algorithm was proposed for the reconstruction phase during inspection in [73]. Views are modeled as vertices and are connected by edges. Each edge is assigned a weight representing the traveling cost for sensor placement planning in [2]; see Fig. 1.4 for an illustration of the approach.

### 1.3 Dissertation scope and contributions

This dissertation studies the object pose estimation problem and the associated view planning problem for data collection. In this regard, it proposes an active robot perception framework whose elements are illustrated in Fig. 1.5. Methods have been developed for each of the components in the proposed framework. The goal is to estimate the object pose to a desired accuracy. The collected data is evaluated to generate potential views, which are informative view directions to collect data in order to improve estimated pose accuracy. The robot motions corresponding to the sensor views are solved by reconciling the potential conflicts between the desired sensor views and constraints from the hardware. The main contributions of this dissertation are:

- A novel plane-based point cloud registration method utilizing convex optimization is pro-

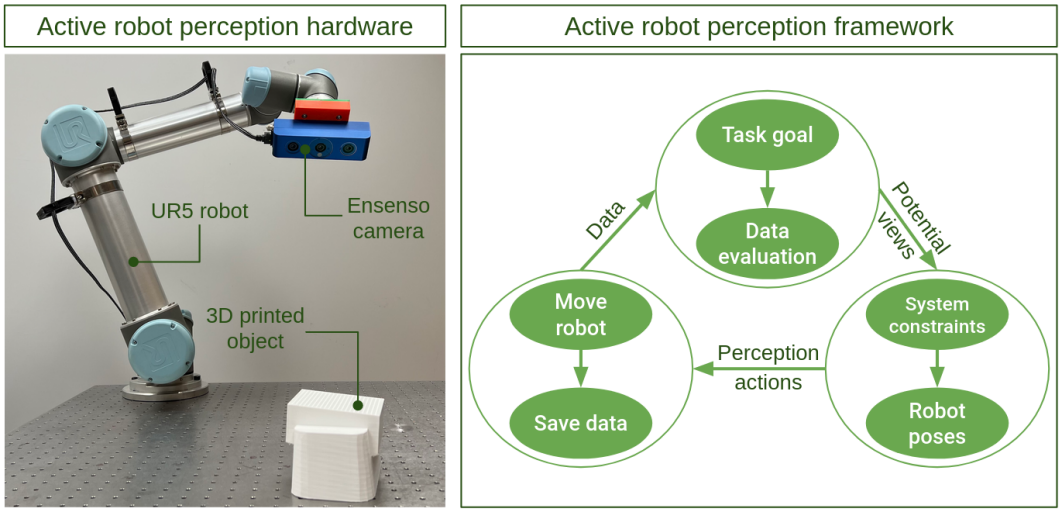



Figure 1.5: The active robot perception framework for 3D object pose estimation

posed.

- A plane-based NBVs strategy for object pose estimation is proposed.
- A point cloud quality predictor based on view angle is proposed.
- Several strategies for sensor view generation are developed based on point cloud quality and quantity analysis. These strategies rely on novel formulations of the set cover and mixed-integer problems for point cloud quality improvement and generating sensor views to increase point cloud quantity
- A constrained nonlinear optimization formulation for solving the inverse kinematics is proposed; this provides the feasibility of achieving robot motion from the current view to the next best view and calculates the robot joint angles required for a given set of sensor views.

## 1.4 Dissertation outline

Chapter 2 provides the preliminaries for the development of the proposed methods, including robot kinematics, point cloud registration, optimization, and motion-force control. Robotic gear chamfering is utilized to illustrate how motion-force control can be employed to reduce workpiece

location uncertainties, which also motivates the development of the active perception methods for object pose estimation. Chapter 3 presents the point cloud registration method based on planar features (plane segments in point clouds). The proposed method formulates the registration as an optimization problem, which can be solved to obtain the rotation and plane-to-plane correspondences. The developed plane-based registration method can estimate the object pose from point cloud registration and provide information for view planning to measure more planes. Chapter 4 presents the proposed NBVs method. Informative directions are defined based on the registration result by utilizing the algorithm proposed in Chapter 3. A view gain is proposed to evaluate view candidates to determine the NBVs to measure useful plane segments for pose estimation. Chapter 5 provides a set of point cloud analysis methods for sensor view generation. The goal is to design methods for objects that do not have planar features. Point clouds are analyzed in terms of quality and quantity to identify the locations in the point cloud that require further measurements and to what extent the point cloud is adequate for registration. The analysis lays a foundation for sensor view generation and view planning, which is discussed in the following chapter. Chapter 6 presents the strategies to generate and plan sensor views based on the analysis from Chapter 5. This chapter focuses on finding the sensor views and associated robot configurations to improve the collected point clouds in terms of quality and quantity. Optimization techniques are utilized to find the minimum number of sensor views and solve inverse kinematics to find the corresponding robot joint angles. Chapter 7 provides the conclusions of this dissertation and future work.

# 2. PRELIMINARIES

This chapter provides the preliminaries for the developed methods in the active robot perception framework, including point cloud processing, robot kinematics, optimization, and motion-force control. Point cloud processing not only provides important information that is used in generating sensor view candidates but also for obtaining the accurate object pose. Robot kinematics is fundamental to calculating the feasible robot configurations for sensor view candidates. Optimization techniques are used to solve multiple problems, including finding the sensor views, registering point clouds, and solving the inverse kinematics. In addition, motion-force control and its implementation in robotic chamfering are briefly discussed; the robotic chamfering application motivated the workpiece localization problem and the active perception approach as a solution to that problem.

## 2.1 Point cloud processing

This section presents the point cloud processing techniques involved in the proposed methods. In order to use the raw point clouds obtained from the vision sensors, some key processing techniques are necessary, such as the normal vector calculation, consistent normal vector orientation, point cloud segmentation, and normal vector clustering (or point cloud clustering based on normal vectors).

### 2.1.1 Calculation and orientation of normal vectors

Normal vectors of a point cloud contain the orientation information of the surfaces from which the points are sampled. Normal vectors have been used in multiple occasions throughout this work, such as finding plane segments in a point cloud, registering two point clouds that contain plane segments, and finding sensor view candidates to measure the object.

The normal vector of a point on a surface is the unit vector that is perpendicular to the surface at that point. One can: (1) reconstruct the surfaces using the collected point clouds and calculate the normal vectors using the constructed surfaces, or (2) estimate the normal vectors directly using

the points. The latter strategy (2) is employed in this work. The normal vector of each point is estimated locally by using a neighborhood of points. The problem is then equivalent to finding the normal vector to a plane fitted using the neighborhood of points, which can be solved by using the Principal Component Analysis (PCA) approach. For a point and its $k$ neighbor points $\mathbf{p}_i$, denote the centroid of the $k$ points as $\bar{\mathbf{p}}$. Then, the covariance matrix of the $k$ points is defined as:

$$\frac{1}{k}\sum_{i=1}^{k}(\mathbf{p}_i - \bar{\mathbf{p}}) \cdot (\mathbf{p}_i - \bar{\mathbf{p}})^T. \tag{2.1}$$

The eigenvector corresponding to the smallest eigenvalue of the covariance matrix is selected as the normal vector to the fitted plane, which is the estimated normal vector of the point. Implementation details, including how to define such a neighborhood, can be found in libraries such as Open3D[1] and PCL[2].

An issue arises when calculating the normal vector using a neighborhood of points if points lie on sharp edges and areas with large curvatures. One can choose to either (1) remove the points that belong to sharp edges prior to normal vector calculation or (2) design algorithms that are robust to sharp edges. One example of detecting the sharp edges in a point cloud is discussed in [74], which is based on the covariance analysis of a neighborhood of points. Denote the three eigenvalues of the covariance matrix as $\lambda_0$, $\lambda_1$, and $\lambda_2$, where $\lambda_0 \leq \lambda_q \leq \lambda_2$. Define the surface variance as:

$$\alpha_k(\mathbf{p}) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{2.2}$$

which is used to determine if a point $\mathbf{p}$ belongs to an edge. Another approach to identifying points that belong to edges is via the use of Gauss map [75], which is based on clustering of the normal vectors on a unit sphere. Normal vectors can be calculated using points that do not belong to sharp edges. The plane segmentation method proposed in this work also has the effect of filtering the points that belong to sharp features. One example of algorithms that are robust to point clouds

[1] Open3D: http://www.open3d.org/

[2] Point Cloud Library (PCL): https://pointclouds.org/

containing sharp features is the method proposed in [76], which is based on a robust version of Randomized Hough Transform (RHT).

Another issue when calculating the normal vectors of a point cloud using the PCA approach is the orientations of the normal vectors. The normal vector of a fitted plane can take either of the two directions that are $180^\circ$ apart if no additional information is given. In this work, the normal vectors of the point clouds are expected to be pointing toward the outside of the object consistently. When the camera location at which the point cloud is collected is available, one can use the camera location to orient the normal vectors. When the camera location is not available, a method referred to as the consistent tangent plane was proposed in [77]. The consistent tangent plane orientation is formulated as a graph optimization problem, which is solved by using an approximation algorithm. In practice, this method is found to be computationally heavy when the point cloud is dense.

### 2.1.2 Clustering using normal vectors

The distribution of the normal vectors of a point cloud on a unit sphere contains important information about the geometry of the object, such as the existence of planar or sharp features. Further, if the object has a convex shape, a point on the object surface and a point on the unit sphere (the normal vector of the point) have one-to-one correspondence. The normal vectors of a plane are mapped to the same point. The following are several techniques that can be utilized to cluster the normal vectors of a point cloud.

*K-means clustering:* k-means clustering is a common method that divides a set of data into k clusters by putting each data point into the cluster with the nearest mean [78]. Since normal vectors distribute on a unit sphere, additional normalization is required when applying k-means to ensure the cluster centroids are still on the unit sphere [79].

*Hierarchical clustering:* Hierarchical clustering seeks to build a hierarchy of clusters [80] which results in a dendrogram. The clustering strategy can be divided into two categories: agglomerative and divisive. The clustering of the point cloud of the three surfaces of a cube by applying Hierarchical agglomerative clustering (HAC) is shown in Fig. 2.1. Notice that for concave objects, HAC may not identify segments correctly. In the next chapter, a method that is based

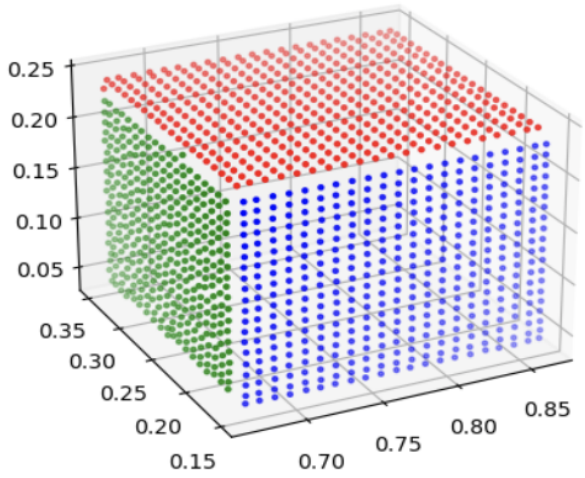



Figure 2.1: Clustering the point cloud of a cube by applying hierachical clustering on the normal vectors.

on Extended Gaussian Image (EGI) is presented.

## 2.2 Robot kinematics

This section provides a brief summary of the kinematics of a robot arm. More detailed and systematic discussions of the kinematics of rigid body mechanisms, representations of position, and orientation can be found in [4, 3, 81].

Figure 2.2 presents an illustration of a robot arm and the robot base frame and the end-effector frame. The forward kinematics of a robot refers to the calculation of the end-effector pose (position and orientation) given the robot joint angles. The inverse kinematics of a robot refers to the calculation of the robot joint angles given the end-effector pose.

### 2.2.1 Forward kinematics

The forward kinematics provides the position and orientation of the end-effector given the joint angles. Denote the robot base frame as $o_b x_b y_b z_b$ and the robot end-effector frame as $o_e x_e y_e z_e$. The position of the end-effector is represented using a vector $\mathbf{p}_e \in \mathbb{R}^3$, and the orientation of the end-effector is represented using a matrix $\mathbf{R}_b^e \in$ SO(3). Further, the position and the orientation of the

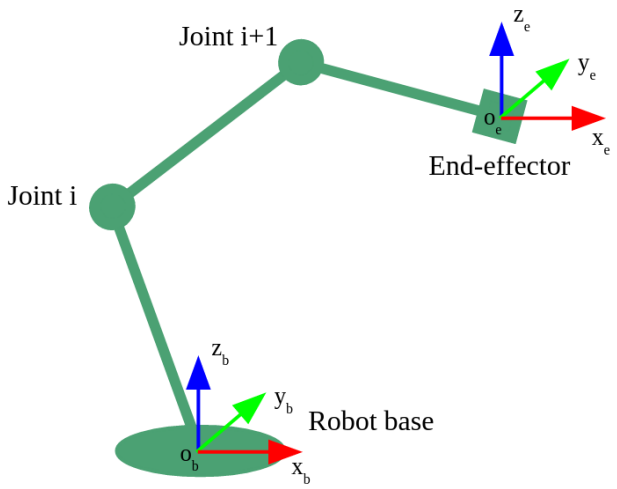



Figure 2.2: An illustration of a robot arm and the robot based and end-effector frames

end-effector can be combined into one homogeneous matrix:

$$\mathbf{T}_b^e = \begin{bmatrix} \mathbf{R}_b^e & \mathbf{p}_b^e \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.3}$$

For a robot arm with $n$ joints, denote the joint angles as $\mathbf{q}, \mathbf{q} \in \mathbb{R}^n$. The forward kinematics refers to the problem of finding $\mathbf{T}_b^e$ given $\mathbf{q}$, or $\mathbf{T}_b^e(\mathbf{q})$.

The calculation of $\mathbf{T}_b^e$ depends on how the coordinate system of the robot is constructed. Two commonly adopted approaches are: the Denavit–Hartenberg (DH) convention [82] and the Product of Exponentials (PoE) formula [3]. The illustrations of the two approaches are shown in Fig. 2.3. When using the DH convention, each robot joint is assigned a coordinate frame following a set of rules. Then, the transformations between adjacent coordinate frames are utilized to calculate the transformation from the robot base frame to the robot end-effector, $\mathbf{T}_b^e$. The PoE formula is based on the exponential coordinate representations of rigid body motions. A fixed coordinate frame is defined at the base of the robot $o_b$, and a frame is attached to the end-effector $o_e$. $\mathbf{T}_b^e$ can be calculated using the transformation between $O_b$ and $O_e$ when all robot joints are at home positions, $\mathbf{M}$, and the joint angles $\mathbf{q}$. The DH convention requires the minimum number of parameters to describe the robot kinematics. For an $n$-joint robot arm, $4n$ parameters are required. The PoE

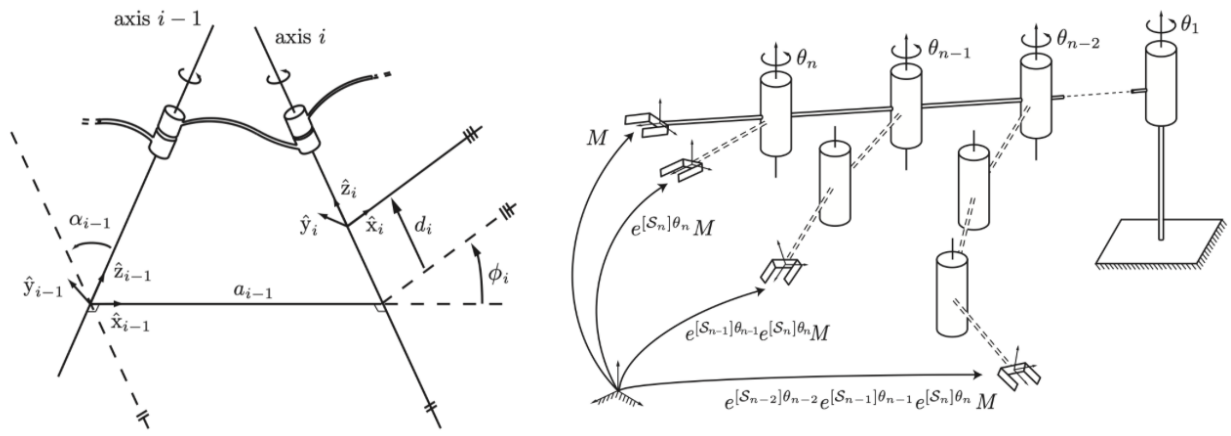



Figure 2.3: Two approaches to construct the coordinate system of a robot arm: using the DH convention (left) and using the PoE formula (right). Figure adopted from [3].

representation requires $6n$ parameters. The PoE approach only requires a home coordinate frame and an end-effector coordinate frame while the DH convention assigns one coordinate frame to each robot joint [3].

### 2.2.2 Inverse kinematics

The inverse kinematics provides the joint angles corresponding to the given position and orientation of the end-effector. Therefore, calculating the inverse kinematics of the robot can be more involved compared to the forward kinematics. Typical questions that are answered are: (1) does there exist a solution? (2) how many solutions? (3) does there exist a closed-form solution?

*Closed-form solution:* Closed-form solutions are usually preferred because the calculation is fast, and all possible solutions can be found. However, it may be difficult to compute the closed-form solutions for most robot configurations. Most industrial robot arms are designed such that closed-form solutions exist for fast inverse kinematics calculation [4]. Closed-form solution methods can be divided into two categories: algebraic and geometric methods. Algebraic methods usually identify the significant equations and manipulate those equations into a solvable form [4]. Geometric methods usually decompose the spatial problem into separate planar problems and solve the individual problems using algebraic manipulations [4]. One widely used software that finds

closed-form inverse kinematics solutions for serial chains is the Ikfast[3].

*Numerical methods:* Oftentimes, numerical methods are used for inverse kinematics since they are not robot dependent, even though numerical methods are slower and usually do not allow computing all solutions [4]. The most common numerical methods include: symbolic elimination methods, continuation methods, and iterative methods [4]. Some commonly used iterative methods include Newton-Raphson [3], nonlinear programming techniques [83], inverse Jacobian methods (including Jacobian transpose, pseudoinverse method, damped least-squares methods) [84]. In this work, the inverse kinematics of the robot needs to be solved to find the robot configurations (joint angles) that satisfy the system constraints while orienting the sensor along the desired view directions.

## 2.3 Optimization

This section presents the concepts of the optimization techniques that have been used in this dissertation.

### 2.3.1 Linear programming

A linear programming problem is an optimization problem that has a linear objective function and linear equality and inequality constraints. Denote the decision variables and parameters of the linear programming problem as $\mathbf{x}, \mathbf{x} \in \mathbf{R}^n$, A, $\mathbf{b}$ and $\mathbf{c}$ are the coefficients of the objective function and the linear inequality constraints. The canonical form of the linear programming problem is the following:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned} \tag{2.4}$$

The linear assignment problem and the set cover problem are two classical problems [85] that can be formulated as linear programming problems with binary or integer decision variables.

*Assignment problem:* The assignment problem is a classical combinatorial optimization prob-

[3]Ikfast: `http://openrave.org/docs/latest_stable/interface_types/`

lem that can be stated in the following general form: A number of agents and a number of tasks are to be matched. Each task can be assigned to at most one agent, and one agent can be assigned to at most one task. Each assignment incurs some *cost*. The goal is to assign as many tasks as possible while minimizing the total cost of the assignments. When the number of the tasks is equal to the number of agents, the problem is called *balanced assignment*. Otherwise, it is called *unbalanced assignment*. The problem is called *linear assignment* if the total cost of all assignments is equal to the sum of the costs of each assignment.

Formally, we are interested in the linear assignment problem: given a set of agents $A = \{\mathbf{a_i}, i = 1, \cdots, L\}$, and a set of tasks $T = \{\mathbf{t_j}, j = 1, \cdots, K\}$. The cost of assigning agent $a_i$ to task $t_j$ is $c_{ij}$. Define the assignment variables $x_{ij}$ such that $x_{ij} = 1$ if agent $a_i$ is assigned to task $t_j$. Find the assignments that minimizes the total cost while satisfying the following constraints:

$$\begin{aligned} \min_{(i,j)\in A\times T} \quad & x_{ij}c_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{K} x_{ij} = 1, \forall i \\ & \sum_{i=1}^{L} 0 \leq x_{ij} \leq 1, \forall j \end{aligned} \tag{2.5}$$

The constraints are such that each agent has only one task, and each task is assigned to at most one agent. In the considered problem, the number of agents is no greater than the number of tasks.

*Set cover problem:* A *set* is a collection of distinct objects, a set of sets is a set, or referred as a *set family*. A *set cover* is a set of subsets whose union has all members of the universe. For a *set system* with the *universe* being $U$ and the set of all subsets being $\mathcal{S}$, i.e., $\cup_{S\in\mathcal{S}} S = U$, each subset is associated with a weight $c : \mathcal{S} \to \mathbb{R}_+$. The task is to find a *set cover* of $(U, \mathcal{S})$ the union of which is equal to the universe, i.e., a subfamily $\mathcal{R} \subset \mathcal{S}$ such that $\cup_{R\in\mathcal{R}} = U$ [85]. As can be seen, the set cover problem can be formulated as an integer linear program. The problem of finding a minimum number of sensor views to cover the target areas on the object is formulated as a minimum weight set cover problem. When the decision variables have a combination of

continuous and integer values, the problem is called a *mixed-integer linear programming* problem, which has the following form:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \leq \mathbf{b} \\ & x_i \in \mathbb{Z}, \text{ for some } i \end{aligned} \tag{2.6}$$

### 2.3.2 Nonlinear programming

A nonlinear programming problem is an optimization problem with at least one of the constraints, or the objective function is nonlinear. Let $n, m, p$ be positive integers and $f(x), h_i(x), g_j(x)$ be real-valued functions. A general nonlinear programming problem has the following form:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0 \quad \forall i \in 1, \cdots, m \\ & h_j(x) = 0 \quad \forall j \in 1, \cdots, p \end{aligned} \tag{2.7}$$

The inverse kinematics problem of calculating the joint angles for a given sensor view candidate is formulated as a constrained nonlinear optimization problem. When some of the decision variables take integer values, the problem becomes a *mixed-integer nonlinear programming* problem. The proposed point cloud registration method by using plane segments is formulated as a mixed-integer nonlinear optimization problem.

## 2.4 Motion-force control

Some manipulation tasks require the application of controlled motions along certain directions as well as controlled forces along other directions. One example of such tasks is that a robot holding a pen is asked to write on a whiteboard. The desired motion in the plane of the whiteboard is specified as well as the desired contact force between the pen and the whiteboard. The general approach to designing a motion-force controller is first presented. The development of a motion-force controller for robotic chamfering of gears is then discussed.

### 2.4.1 Motion-force controller

The basic idea in designing a motion-force controller is to design controllers for desired motions and forces separately. Denote the number of Cartesian degrees-of-freedom (DOF) as $N$, an $N$-tuple vector that selects which Cartesian DOF are under force control as $\mathbf{S}$ (by setting the element in the tuple as 1). The control signal for the $i$-th joint $\tau_i$ is [86]:

$$\tau_i = \sum_{j=1}^{N} \{\Gamma_{ij}(s_i \Delta f_j) + \Psi_{ij}(1 - s_i)\Delta x_j\} \tag{2.8}$$

where $\delta f_j$ is the force error in the $j$-th DoF, $\delta x_j$ is the position error in the $j$-th DoF, $\Gamma_{ij}$ and $\Psi_{ij}$ are the functions to generate control signals using the force and position errors, respectively, and $s_i$ is the $i$-th element in the selection vector $\mathbf{S}$. Control laws such as computed torque, PID can be applied to generate the control signals.

### 2.4.2 Implementation in robotic gear chamfering

Gear chamfering is the process of removing sharp edges on gear teeth for both safety and increasing the funtionality of the gears. Currently, specialized gear chamfering machines exist. But the chamfering is done manually for large size gears due to the limitations in the workspace of the chamfering machines.

The accurate location of the gear in the robotic work cell is required in order to obtain even chamfers with desired sizes. However, uncertainties in the location of gears with respect to the robot always exist in practice due to how the gear is mounted in the work cell fixture. A novel chamfering method, duel-edge chamfering, based on a hybrid motion-force control to identify the gear center and gear root positions is proposed in [87]. Based on this identification, we employed a novel force/motion strategy that can simultaneously chamfer two edges of the adjacent gear teeth. This section briefly summarizes the designed hybrid motion-force controller for the gear root identification*.

*Reprinted with permission from "Dual-edge robotic gear chamfering with registration error compensation" by Jie Hu, Prabhakar Pagilla, 2021, Robotics and Computer-Integrated Manufacturing, 69, 102082. ©2021 by Elsevier.

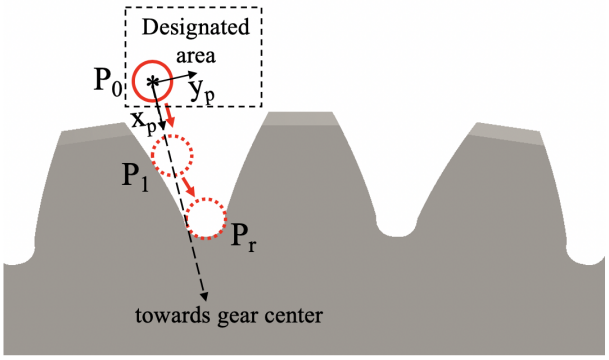



Figure 2.4: Identification of the gear root

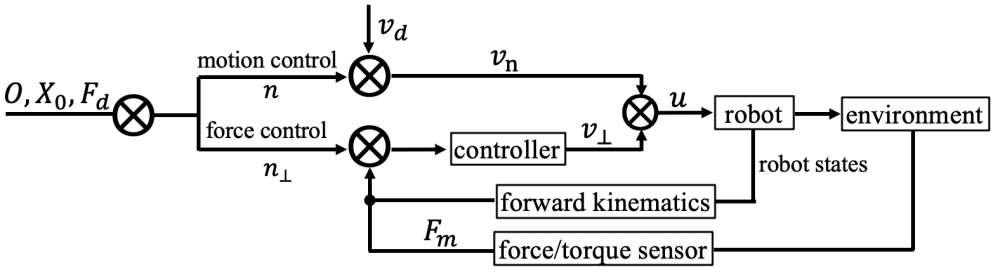



Figure 2.5: Motion/force control block diagram for gear root identification

For the control of the gear root identification process as illustrated in Fig. 2.4, motion and force are controlled simultaneously. The control block diagram in Fig. 2.5 broadly illustrates the motion/force control strategy employed for gear root identification. The initial motion direction ($\mathbf{n} \in \mathbb{R}^3$) and the force control direction ($\mathbf{n}_\perp \in \mathbb{R}^3$) are determined by the initial Cartesian position $\mathbf{X_0}$ (denoted as point $P_0$ in Fig. 2.4) of the end-effector in the designated area and the gear center position $\mathbf{O}$, where $\mathbf{n}$ and $\mathbf{n}_\perp$ are unit vectors along $x_p$ and $y_p$, respectively. Referring to Fig. 2.4, while the end-effector moves towards the gear center along $x_p$ direction (which points to the gear center) at a constant velocity ($v_d$) during the gear root identification, the motion perpendicular to $x_p$ is regulated by the force control law with zero as the force setpoint. Force control is activated once the end-effector makes contact with a gear edge and the force sensor registers a non-zero contact force. Once the contact is established at Point $P_1$, the desired motion of the end-effector is

along the line joining $P_1$ and the gear center. A hybrid impendance type force control is employed so that the motion evolves along the gear edge; the force control law utilizes proportional action on three measured variables along the $y_p$ direction: position error, velocity error, and force error.

The detailed strategy to calculate the control input $\mathbf{u}$ (robot joint velocity vector) is provided in **Algorithm** 1. The gear root is found when the resultant contact force reaches at the preset threshold $f$. When the end-effector slides on the gear tooth contour, the desired Cartesian velocity of the end-effector $\dot{\mathbf{x}}$ is calculated by using three parts, $\mathbf{v_n}, \omega$, and $v_{ctrl} \in \mathbb{R}$. The direction of $\mathbf{v_n}$ is along the nominal motion direction with magnitude of $v_d$. $v_{ctrl}$ is the velocity scalar along the perpendicular motion direction, and is calculated according to the control law by using the current force error, displacement, and velocity of the end-effector that are projected on the perpendicular direction. The states of the robot such as the joint positions and joint velocities are accessed from the robot controller to calculate the end-effector positions and velocities $\mathbf{X}$, $\dot{\mathbf{X}}$ in Cartesian space. The control input $\mathbf{u}$ to the robot is calculated using the current Jacobian $\mathbf{J}$ and the desired Cartesian velocity input $\dot{\mathbf{x}}$. More detailed discussions and experiment results can be found in [87, 88].

**Algorithm 1:** Gear root identification

**Input:** $\mathbf{X_0}, \mathbf{O}, \mathbf{X}, \dot{\mathbf{X}}, v_d, F_d, T, k_d, k_v, k_f, f$
**Output:** $\mathbf{X_{root}}, R_{base}^{root}$
Initialization: $\mathbf{n} = \frac{\overrightarrow{X_0O}}{\|X_0O\|}$, $\mathbf{n}_\perp \cdot \mathbf{n} = 0$, $\omega = \mathbf{0}$,
$a = 0, v_{ctrl} = 0$;
**while** $\| \mathbf{F_m} \| < f$ **do**
  update $\mathbf{X}, \dot{\mathbf{X}}, \mathbf{F_m}$;
  $\mathbf{v_n} \leftarrow v_d \cdot \mathbf{n}$;
  $a \leftarrow -k_d(\mathbf{X} - \mathbf{X_0}) \cdot \mathbf{n}_\perp - k_v \dot{\mathbf{X}} \cdot \mathbf{n}_\perp + k_f(\mathbf{F_m} \cdot \mathbf{n}_\perp - F_d)$;
  $v_{ctrl} \leftarrow v_{ctrl} + aT$;
  $\mathbf{v}_\perp \leftarrow v_{ctrl} \cdot \mathbf{n}_\perp$;
  $\dot{\mathbf{x}} \leftarrow [\mathbf{v_n} + \mathbf{v}_\perp, \omega]$;
  $\mathbf{u} = \mathbf{J}^{-1}\dot{\mathbf{x}}$;
  maintain loop time($T$);
**end**
$\mathbf{X_{root}} \leftarrow \mathbf{X}$; $R_{base}^{root}$

# 3. PLANE-BASED POINT CLOUD REGISTRATION

This chapter presents a point cloud registration method that utilizes plane segments in the point clouds. The motivation comes from the observation that many engineered parts have planar features and that geometric features such as planes can serve as natural connections between pose estimation and view planning. The plane-based next-best-view, which will be discussed in the next chapter, utilizes the registration results to generate and select sensor views. The problem is first formulated, and an overview of the method is provided. A novel plane segmentation method is presented. A convex optimization problem is formulated to solve the plane-to-plane correspondence and the rotation matrix between two point clouds. The translation is solved using different methods for convex and nonconvex-shaped objects. The proposed registration method is further compared with the state-of-the-art registration algorithms. Simulation and experimental results are provided*.

## 3.1 Problem description and method overview

Consider the workpiece shown in Fig. 3.1, where the frames corresponding to the robot base, vision sensor, nominal workpiece, and actual workpiece, are given by $C_w$, $C_s$, $C_c$, and $C_a$, respectively, and their corresponding origins are given by $O_w$, $O_s$, $O_c$, and $O_a$. Our goal is to obtain the workpiece pose by finding the homogeneous transformation from $O_c$ to $O_a$. We assume the workpieces have at least three planes whose normal vectors are neither co-planar nor parallel. We use plane segments to localize workpieces because engineered workpieces in manufacturing usually have plane segments, and the transformation can be calculated for two sets of plane segments. Both the CAD model point cloud and the measured point clouds from the vision sensor are processed to extract plane segments. While the CAD model point clouds can be segmented offline, the measured point clouds are segmented online sequentially. In order to find the transformation

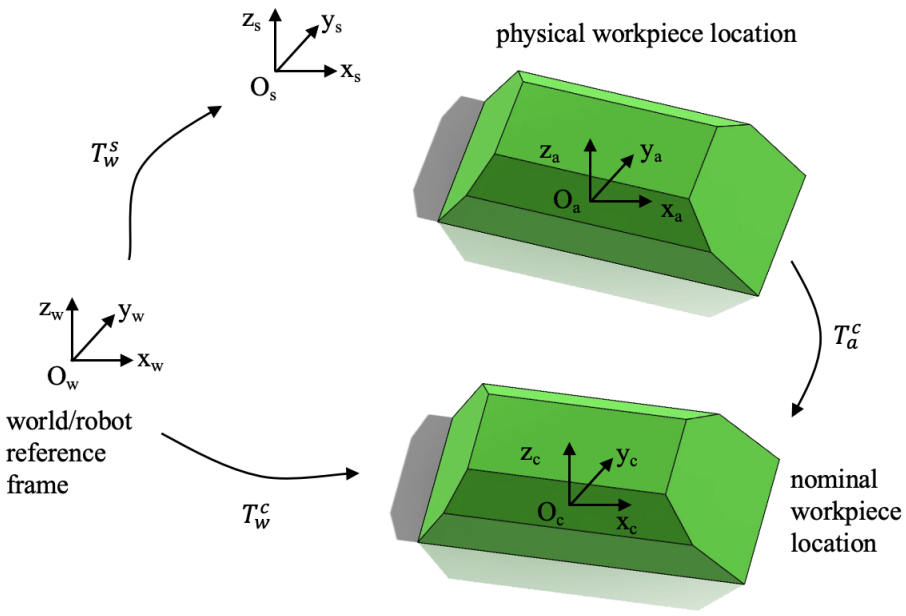



Figure 3.1: Coordinate frames of the object pose estimation problem through registration

from $O_c$ to $O_a$, one needs to find the plane-to-plane correspondence between measured plane segments and the CAD model plane segments, that is, for each plane segment in the set $S'$, find the corresponding plane segment from the set $S$.

The proposed registration method includes three main tasks: extracting plane segments, finding the correspondence between the measured plane segments and the CAD model plane segments, calculating the transformation between two point clouds. The calculation of the rotation and translation is decoupled. For the development of the methods, the workpiece is assumed to be convex shape. Adaptations to nonconvex shape workpieces are further discussed. The plane segments in the point clouds are first identified and extracted using the proposed method. Each plane segment can be represented by the normal vector, centroid, and the signed distance to the coordinate origin. An optimization problem is formulated and solved to find the correspondence along with the rotation between the point clouds. For the calculation of the translation for convex shape workpieces utilizes Singular Value Decomposition (SVD). For nonconvex shape workpieces, a two-step registration based on ICP is applied.

### 3.2 Extracting plane segments from point cloud

In this section, we present a method to identify the plane segments in a point cloud and extract the key information to represent such plane segments. The method is used to process both the measured point clouds and the point cloud sampled from the CAD model. The key idea is to project the normal vectors of a point cloud on a unit sphere. Since the points on a plane segment have the same normal vector when there is no computation errors, each plane segment is mapped into a point on the unit sphere. Considering the errors in calculating the normal vectors and measurement errors, a plane segment will be mapped into a compact area on the unit sphere. Extracting plane segments is equal to identify such compact areas on the unit sphere.

**Algorithm 2:** Normal vectors clustering

```
Result: Rg = {rg_i}
Initialization: δ = C, rsl_l, rsl_u, Rg, Rg' = ∅;
while δ ≠ 0 do
    for rsl = rsl_l; rsl < rsl_u; rsl = rsl + 1 do
        for pix ∈ Pixels do
            nbs ← getNeighbors(pix);
            if any(nbs) ∈ Rg then
                largest_region(Rg)←pix;
            else
                Rg←new_region(pix)
            end
        end
        δ = ||Rg| − |Rg|'|;
        Rg' =Rg;
    end
end
```

The key steps of the proposed method are the following: (i) calculating normal vectors for every point using the k-nearest-neighbor method; (ii) project the normal vectors to a unit sphere and find the clusters using the proposed method; (iii) characterize the size of each cluster and find the compact clusters that are considered to represent plane segments. Many open source libraries

have implementations for the normal calculation of a point cloud that can be used in Step (i). In this work, we use the implementation provided in Open3D library [89]. For Step (ii), one needs to discretize the unit sphere into patches with equal areas, and merging adjacent patches to represent the mapped normal vectors that belong to a surface region on the object. The discussion of the discretization or tessellation of unit sphere can be found in [90]. In this work, HEALPix[1] [91] library is used to discretize the unit sphere into curvilinear quadrilaterals (referred to as the pixels in the library) of equal sizes. The size of the pixel can be controlled by setting the resolution parameter in the software. In this work, the resolution parameter is increased gradually until the obtained cluster number stops changing to avoid inappropriate clustering due to the selection of resolution. For each resolution, we first filter out the pixels that have normal vectors less than a threshold, which are considered as noises. Adjacent areas are merged to form multiple regions. The merging process is described in Algorithm 2. In Algorithm 2, $Rg$ is the set that includes all the regions $rg_i$ on the unit sphere, $rsl_l, rsl_u$ are the lower and upper bound of the resolution, $Rg'$ is the previous cluster set, $\delta$ is the size difference between the obtained sets, $Pixels$ is the set of all the pixels at a given resolution $rsl$. The algorithm stops when the obtained cluster number stops increasing.

To characterize the sizes of the obtained regions, we fit a plane using the normal vectors of each cluster, and look at the intersection between the plane and the unit sphere. For a plane segment on a workpiece, the cluster of the normal vectors should be compact, i.e., the intersection of the fitted plane and the unit sphere should have small area, which is the criterion we have adopted in this work to identify plane segments in a point cloud. Two cases with difference intersection areas are shown in Fig. 3.2

Once the clusters on the unit sphere that correspond to the plane segments in the point cloud are identified, the points in the point cloud that correspond to the normal vectors in each cluster can be extracted by using the indices of the points. The normal vector, centroid, and the signed distance can also be calculated by using the extracted points that represent a plane segment. Next, we

[1]HEALPix: http://healpix.sourceforge.net

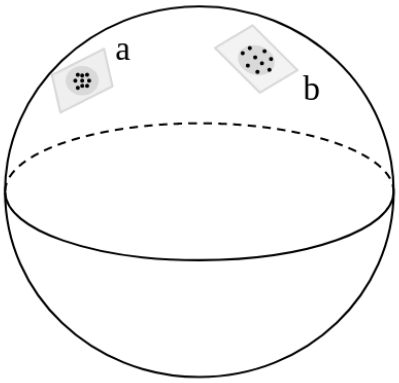



Figure 3.2: Two cases with difference intersection areas when characterizing normal clusters on a unit sphere. The clustered normal vectors in (a) are more compact comparing to (b).

discuss the method to find the plane-to-plane correspondence using the identified plane segments and the normal vectors.

### 3.3 Rotation calculation through convex optimization

Assuming a set of normal vectors corresponding to the plane segments from the measured point cloud is: $N' = \{\mathbf{n}'_i, i = 1, \cdots, K\}$, and the set of normal vectors from the CAD model point cloud is $N = \{\mathbf{n}_j, j = 1, \cdots, L\}$, and $L \geq K$. The problem of finding the correspondence for each $\mathbf{n}'_i$ from $N$ and calculating the rotation matrix $\mathbf{R} \in$ SO(3) that rotates $N$ to $N'$ can be formulated as the following problem. Define a binary variable $X_{ij}$ to indicate the correspondence relation between $\mathbf{n}_j \in N$ and $\mathbf{n}'_i \in N'$. $X_{ij} = 1$ indicates that $\mathbf{n}_j$ is the correspondence to $\mathbf{n}'_i$, and 0 otherwise. Then we have: $\mathbf{n}'_i = \mathbf{R}\sum_{j=1}^{K} X_{ij}\mathbf{n}_j$. The solution to the following optimization problem gives $\mathbf{R}$ and $X_{ij}$:

$$\min_{\mathbf{R}, X_{ij}} \sum_{i=1}^{L} \|\mathbf{n}'_i - \mathbf{R}\sum_{j=1}^{K} X_{ij}\mathbf{n}_j\|^2 \tag{3.1}$$

subject to the following constraints:

$$X_{ij} \in \{0, 1\} \tag{3.2}$$

$$\sum_j X_{ij} = 1, \forall i \tag{3.3}$$

$$\sum_i X_{ij} \leq 1, \forall j \tag{3.4}$$

$$\mathbf{R} \in \text{SO(3)} \tag{3.5}$$

Each normal vector of the measured plane segment should have and only have one correspondence from the CAD model plane segments normals, and each of the normals from the CAD model can have at most one correspondence in the measured plane normals. These two constraints are reflected by Eq. (3.3) and (3.4). Further, the above optimization problem is non-convex due to the following reasons: (1) the constraint that $\mathbf{R} \in$ SO(3) is non-convex, and (2) the product of $X_{ij}$ with the rotation matrix $\mathbf{R}$ causes non-convexity. In order to make the problem tractable, we use the following two relaxations: (1) relax the constraint that $\mathbf{R} \in$ SO(3) to be that $\mathbf{R}$ is in the convex hull of SO(3) matrices (indicated by conv SO(3)), and (2) lifting the product of $X_{ij}$ and $\mathbf{R}$ using the McCormick relaxation [92]. The resulting formulation is the following:

$$\min_{\mathbf{A}, X_{ij}} \sum_{i=1}^{L} (2 - 2(\mathbf{n}_i')^T \sum_{j=1}^{K} \mathbf{M}_{ij} \mathbf{n}_j) \tag{3.6}$$

subject to the following constraints:

$$-X_{ij} \leq m_{pq}^{ij} \leq X_{ij} \tag{3.7}$$

$$r_{pq} + X_{ij} - 1 \leq m_{pq}^{ij} \leq r_{pq} - X_{ij} + 1 \tag{3.8}$$

$$\mathbf{A} \succeq 0 \tag{3.9}$$

$$X_{ij} \in \{0, 1\} \tag{3.10}$$

$$\textstyle\sum_j X_{ij} = 1, \forall i \tag{3.11}$$

$$\textstyle\sum_i X_{ij} \leq 1, \forall j \tag{3.12}$$

$$\mathbf{A} = \begin{bmatrix} 1 - r_{11} - r_{22} + r_{33} & r_{13} + r_{31} & r_{12} - r_{21} & r_{23} + r_{32} \\ r_{13} + r_{31} & 1 + r_{11} - r_{22} - r_{33} & r_{23} - r_{32} & r_{12} + r_{21} \\ r_{12} - r_{21} & r_{23} - r_{32} & 1 + r_{11} + r_{22} + r_{33} & r_{31} - r_{13} \\ r_{23} + r_{32} & r_{12} + r_{21} & r_{31} - r_{13} & 1 - r_{11} + r_{22} - r_{33} \end{bmatrix} \tag{3.15}$$

where:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

and $\mathbf{M}_{ij}$ is defined as:

$$\mathbf{M}_{ij} = \begin{bmatrix} r_{11}X_{ij} & r_{12}X_{ij} & r_{13}X_{ij} \\ r_{21}X_{ij} & r_{22}X_{ij} & r_{23}X_{ij} \\ r_{31}X_{ij} & r_{32}X_{ij} & r_{33}X_{ij} \end{bmatrix}$$

For each element $m_{pq}^{ij}$ in $\mathbf{M}_{ij}$: $m_{pq}^{ij} = r_{pq}X_{ij}, p, q \in \{1, 2, 3\}$, we apply the McCormick relaxation based on the two constraints:$-1 \leq r_{pq} \leq 1$, $0 \leq X_{ij} \leq 1$. we have:

$$-X_{ij} \leq m_{pq}^{ij} \leq X_{ij} \tag{3.13}$$

$$r_{pq} + X_{ij} - 1 \leq m_{pq}^{ij} \leq r_{pq} - X_{ij} + 1 \tag{3.14}$$

Matrix $\mathbf{A}$ in Eq.(3.15) is defined based on the definition of the convex hull of SO(3) in [93]: conv SO(3) $= \{\mathbf{R} \in \mathbb{R}^{3\times3} : \mathbf{A}^{4\times4} \succeq 0\}$. The reformulated optimization problem is easier to solve for a lower bound of the original non-convex problem. After solving the correspondence problem, the rotation $\mathbf{R}$ is also known. A solution to the relaxed problem may not always result in $\mathbf{R} \in$ SO(3); a simple projection to SO(3) can be performed using the solution of Orthogonal Procrustes problem [19].

### 3.4 Translation calculation

The translation remains to be calculated for the two point clouds to be registered. For objects with convex shapes, the normal vectors of each plane segment is projected onto the unit sphere as a separate cluster, i.e., the clusters on the unit sphere and the plane segments in the object have one-to-one correspondence. Thus, we can use the solved correspondence and the plane information to calculate the translation. Assuming the signed distances for the obtained measured plane segments are $d_i'$, and $d_j$ for the plane segments from the CAD model, then the following least-squares problem is solved by using Singular Value Decomposition (SVD) to calculate the translation $\mathbf{t}$ between the CAD model data and the measured data:

$$\min_{\mathbf{t}\in\mathbb{R}^3} \sum_{i=1}^{L} \|d_i' - (\sum_{j=1}^{K} X_{ij} d_j + \langle \sum_{j=1}^{K} X_{ij}\mathbf{n}_j, \mathbf{t}\rangle)\|^2 \tag{3.16}$$

### 3.5 Extension to nonconvex objects

When the objects are not convex, the one-to-one correspondence between the normal vector clusters on the unit sphere and the plane segments on the object does not exist. Plane segments that have the same or close normal vectors could be projected onto the same or adjacent pixels on the unit sphere, which can lead to the clustering of different plane segments into one. The incorrect plane segmentation will also generate erroneous rotations and translations. For this type of object, the following registration strategy is used: applying the proposed strategy to extract plane segments and solve the rotation, bringing the measured point cloud and the CAD model point cloud to coarse alignment utilizing the solved rotation and applying translation based on the centroid difference between the point clouds. ICP is applied to bring the point clouds to fine alignment.

### 3.6 Simulation and experimental results

The following results are presented: effectiveness of extracting plane segments using the proposed method for both concave and convex objects, the comparison of registration results of the

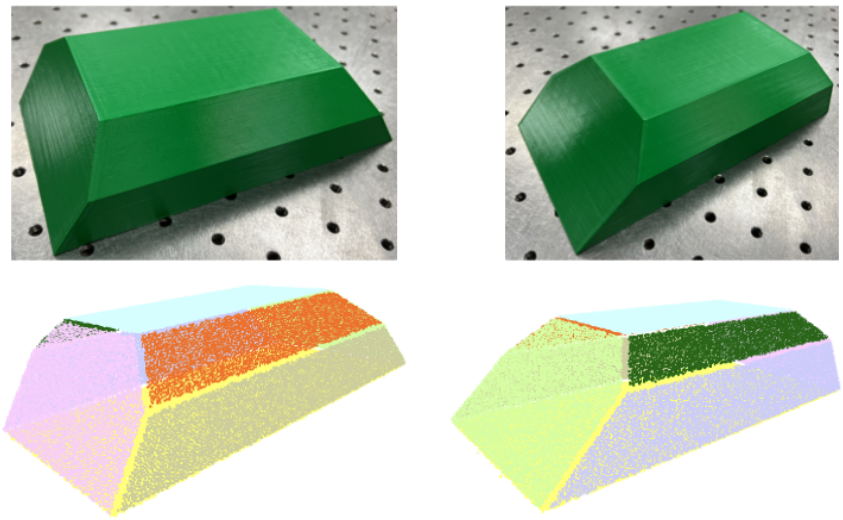


Figure 3.3: Extracted plane segments of a convex object by applying the proposed method. Top: two views of the convex object. Bottom: two views of the plane segments shown in different colors.

proposed method and two state-of-the-art registration algorithms: Go-ICP [94] and Fast Global Registration (FGR) [95]. Some of the results are published in [36].

### 3.6.1 Plane segments identification

Figure 3.3 shows two views of the extracted plane segments by applying the proposed algorithm. The object has eight planes in total. The extracted planes are highlighted in different colors. As shown in the figure, all plane segments have been identified correctly. Additionally, it is observed that the points on the edges are removed during the clustering process. The reason is that the normal vectors on the edges are either clustered into the pixels of the adjacent plane segments, or mapped onto pixels that are considered as noises and filtered due to the small number of normal vectors. A similar finding is reported in [75], in which Gauss map is utilized to detect sharp edges.

Figure 3.4 and Fig. 3.4 are two examples when applying the algorithm to concave objects. These object models are obtained from Fusion360 dataset[2] and MCB dataset[3]. The object in Fig. 3.4 has several plane segments that are parallel, as numbers in the figure. The parallel planes

[2]Fusion360: https://github.com/AutodeskAILab/Fusion360GalleryDataset
[3]MCB: https://mechanical-components.herokuapp.com

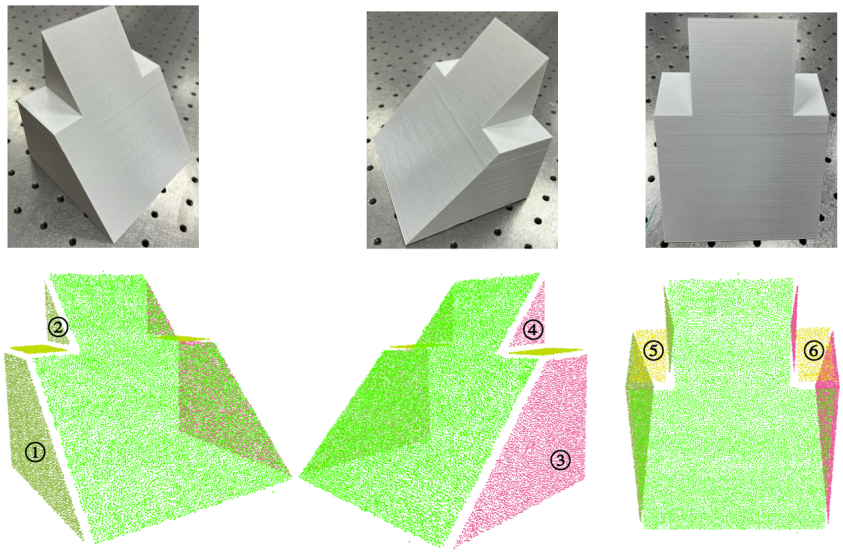



Figure 3.4: After applying the proposed normal clustering method, the obtained normal vectors clusters may contain the normal vectors that belong to multiple parallel planes, such as the numbered 1 and 2 plane segments, 3 and 4 plane segments, and 5, 6 plane segments.

are thus identified as the same cluster, which are highlighted using the same color. Figure 3.5 shows another example of the segmentation of a concave object. For applications in which parallel planes have to be distinguished and separated, segmentation based on Euclidean distance can be further applied into the pipeline. However, the proposed segmentation method aims to find the correspondences and rotation between two point clouds. The rotation can still be calculated correctly even when the normal vectors of one region belong to multiple plane segments, as long as the same clustering process is applied to both the CAD point cloud and the measured point cloud. Notice that when symmetry exists, the rotation will be ambiguous, but the ambiguity also exists for many existing point cloud registration algorithms. Registration results are included in the following section.

### 3.6.2 Registration comparison

This section provides the registration comparison of the proposed plane-based registration and two other state-of-the-art registration algorithms: Go-ICP [94] and Fast Global Registration (FGR) [95] using both convex objects and concave objects.

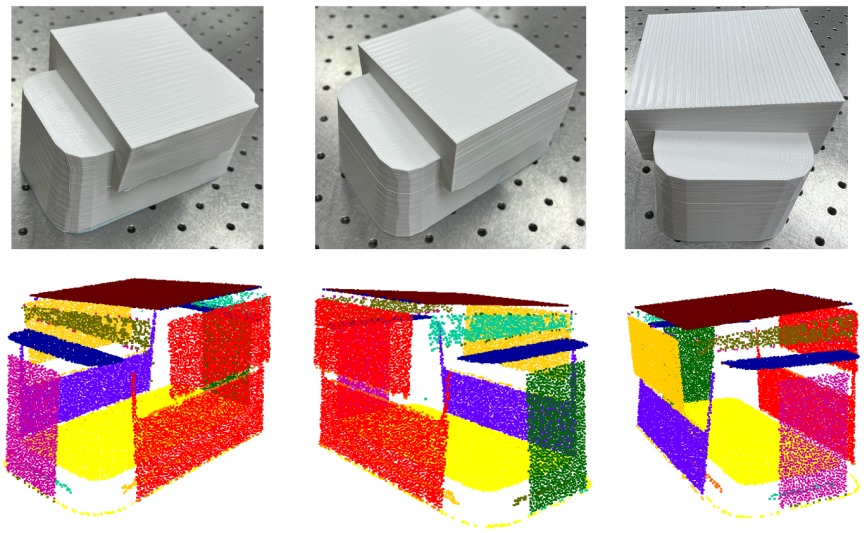


Figure 3.5: Parallel plane segments that have the same normal vectors fall into the same clusters on the unit sphere.

We first present the registration results for a convex object. The rotation and translation are calculated separately in the proposed method. A sample of registration results is provided. The registration results are analyzed using two metrics: fitness and inliers Root Mean Square Error (RMSE). Fitness is defined as the ratio of the number of inliers to the total points. The inlier RMSE is calculated based on the Euclidean distances between the inliers and their correspondences. Point cloud 1-3 are partial point clouds sampled from the entire point cloud, as shown in Fig. 3.6. Point cloud 4-6 are point cloud 1-3 with 3D Gaussian noise. The standard deviation of the Gaussian distribution is set to 0.003. Five trials with random transformations are conducted. A random rotation is obtained by generating a random rotation axis (a unit vector) and a random angle in the range of $[-\pi, \pi]$. A random translation is obtained by generating a random 3D vector, each element is in the range of $[-0.5, 0.5]$ m. The average fitness, RMSE and time are reported in Table 3.1.

For point cloud 1-3 (data 1-3 in Table 3.1) that have no noises, FGR outperforms the proposed method and GO-ICP in terms of computation time and RMSE. The proposed method has higher fitness and lower computation time comparing to Go-ICP. Comparing to FGR, the proposed method has higher computation time but equal registration fitness. When noises are added to the

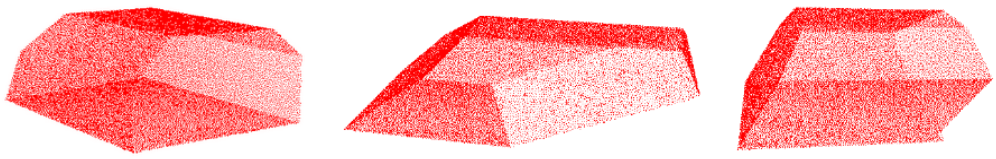


Figure 3.6: Partial point clouds of a convex object used in the registration comparison

Table 3.1: Registration comparison a using convex object in terms of RMSE (mm) and time (s)

| Data | Go-ICP | | | FGR | | | Proposed | | |
|---|---|---|---|---|---|---|---|---|---|
| | Fitness | RMSE | Time | Fitness | RMSE | Time | Fitness | RMSE | Time |
| 1 | 0.885 | 0.91 | 431.24 | 1.0 | 0.003 | 3.36 | 1.0 | 0.34 | 5.86 |
| 2 | 0.739 | 1.28 | 18.17 | 1.0 | 0.018 | 3.17 | 1.0 | 0.11 | 5.44 |
| 3 | 0.084 | 1.27 | 159.91 | 1.0 | 0.002 | 3.49 | 1.0 | 0.21 | 5.87 |
| 4 | 0.823 | 1.11 | 29.34 | 0.388 | 1.36 | 3.66 | 0.997 | 0.86 | 6.99 |
| 5 | 0.988 | 1.08 | 20.37 | 0.519 | 1.37 | 4.20 | 0.986 | 1.09 | 7.85 |
| 6 | 0.806 | 1.20 | 331.31 | 0.060 | 1.38 | 4.26 | 0.995 | 1.0 | 7.57 |

point clouds (data 4-6 in Table 3.1), the proposed method maintains high fitness ($> 0.98$) with slightly increased computation time comparing to registering using point clouds without noises. The registration fitness when using FGR is decreased significantly even though the computation is faster than the proposed method and Go-ICP. Registration using Go-ICP has higher fitness when compared to FGR but with significantly longer computation time.

We also compare the proposed plane-based point cloud registration method with Go-ICP and FGR using two sets of data from two nonconvex objects, which are shown in Fig. 3.8. The sensor data is collected offline using the vision sensor Ensenso N35 which is mounted on a UR5 robot. The proposed method for nonconvex objectis a two-step coarse-to-fine registration. Figure 3.7 provides an example of the two steps. After registration, fitness and RMSE are utilized to evaluate the registration results.

Table 3.2 includes the registration results when using point clouds of the two concave objects. Data 1-3 are the point clouds from the object shown in the first row in Fig. 3.8. Data 4-6 are

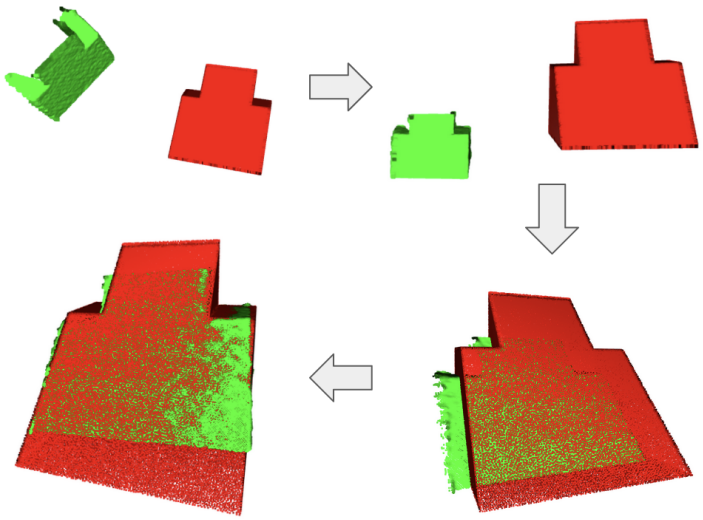


Figure 3.7: The proposed coarse-to-fine registration method. Top left: the initial two point clouds, red is the CAD point cloud, green is the measured point cloud. Top right: point clouds after applying the obtained rotation. Bottom right: point clouds after translating the measured point cloud by the centroid difference. Bottom left: the result after applying ICP.

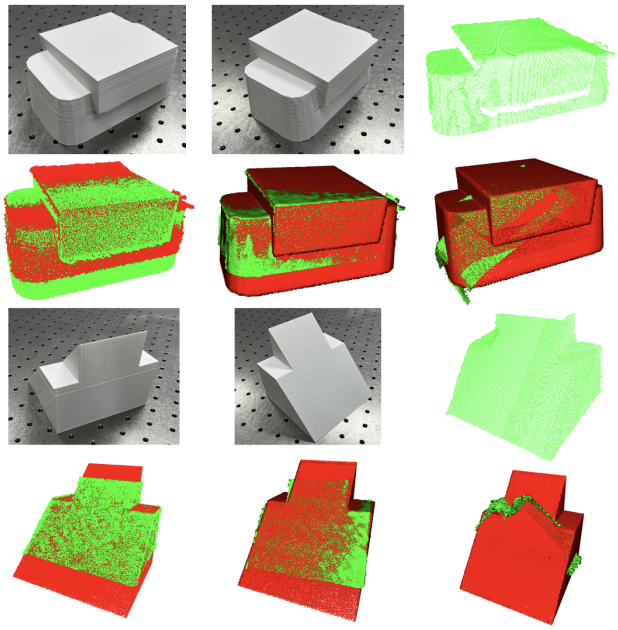


Figure 3.8: Top: the complete(red) and partial(green) point clouds of the objects used in the comparison. Bottom: point clouds after registration using Go-ICP (left), our method (middle), and FGR (right).

Table 3.2: Registration comparison using concave objects in terms of RMSE (mm) and time (s).

| Data | Go-ICP | | | FGR | | | Proposed | | |
|---|---|---|---|---|---|---|---|---|---|
| | Fitness | RMSE | Time | Fitness | RMSE | Time | Fitness | RMSE | Time |
| 1 | 0.701 | 0.64 | 79.77 | 0.202 | 1.74 | 1.62 | 0.786 | 1.16 | 2.25 |
| 2 | 0.666 | 0.65 | 70.77 | 0.197 | 1.77 | 1.88 | 0.721 | 1.14 | 1.47 |
| 3 | 0.959 | 0.59 | 33.82 | 0.174 | 1.79 | 2.17 | 0.864 | 0.92 | 2.80 |
| 4 | 0.865 | 0.60 | 233.98 | 0.510 | 1.53 | 2.02 | 0.872 | 1.12 | 1.26 |
| 5 | 0.877 | 0.59 | 63.80 | 0.348 | 1.78 | 1.54 | 0.804 | 1.21 | 0.96 |
| 6 | 0.878 | 0.59 | 68.26 | 0.284 | 1.82 | 1.32 | 0.888 | 1.05 | 0.98 |

the point clouds from the object shown in the third row in Fig. 3.8. For both sets of data, our registration method is faster than Go-ICP, has lower registration errors than FGR.

## 3.7 Conclusions

In this chapter, a plane-based registration method is developed. The benefits of the proposed registration method are twofold: (1) provide a new point cloud registration approach based on plane segments that utilizes convex optimization, and (2) provide an interface for plane-based NBVs for data collection. The proposed plane segmentation methods based on normal vector clustering on the unit sphere are proven to be effective for convex objects in the sense that planar features on the object can be detected, and the point clouds can be segmented correctly. The segmentation method can also be applied to nonconvex objects with the caveat of clustering plane segments that have the same or close normal vectors into the same segment. The proposed segmentation method is still applicable with the proposed registration method if the segmentation is applied to both the point cloud from the CAD model and the measured point cloud. The next chapter discusses how the registration result can be used to define informative sensor views to measure more plane segments to improve registration accuracy.

# 4. PLANE-BASED NEXT-BEST-VIEW FOR OBJECT POSE ESTIMATION

This chapter presents one next-best-view method to measure planes on the object for pose estimation, which uses the plane-based registration result that developed in the previous chapter. The goal of the developed view planning method is to find sensor view directions that lead to the measure of more plane segments in order to increase the point cloud registration accuracy. The problem and method overview is first provided. Determining the NBVs consists of two components: (1) defining informative view directions and sensor view candidates, and (2) selecting from the sensor view candidates the NBVs based on the proposed criterion. Simulation and experimental results are provided to show the effectiveness of determining informative view directions, generating sensor view candidates, finding the NBVs, and the improved point cloud registration*.

## 4.1 Problem description and method overview

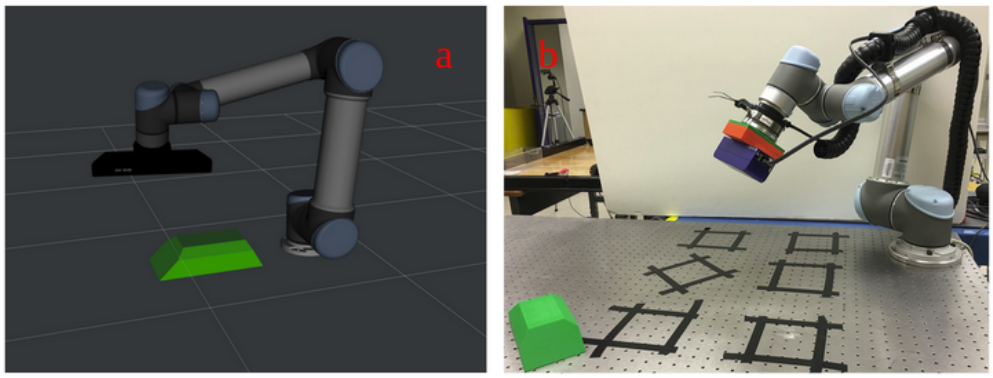



Figure 4.1: System setup: (a) simulation using ROS; (b) experimental setup. Vision sensors are mounted on the robot end-effector. A 3D printed object is placed in the workspace.

The considered robotic system setup is shown in Fig. 4.1. The sensor is mounted on the robot end-effector. The object is placed in the robot workspace to be localized. At least three planar

*©2021 IEEE. Part of this chapter is reprinted with permission from Jie Hu, Prabhakar Pagilla, Swaroop Darbha, "A Novel Method for the Localization of Convex Workpieces in Robot Workspace Using Gauss Map", IEEE Conference on Decision and Control, December, 2021.

features that are parallel exist on the object. The robot arm places the sensor at multiple locations to collect point clouds by measuring the object. Collected point clouds are utilized to estimate the object pose by using the proposed registration method discussed in the previous chapter. The next-best-view problem in this context is to find the next best sensor view that can measure more planar features on the object such that the plane-based registration results can be improved.

The proposed method for the NBV problem is decomposed into the following three steps: (1) determination of informative sensor views, (2) determination of possible view directions and positions (Cartesian location of the sensor), and (3) NBV selection from among the possible views. To address (1), the extracted plane segments from both the measured point clouds and the CAD model point cloud, along with the current registration result, are utilized. In order to find potential sensor view directions and sensor positions, the robot workspace is discretized into voxels. Each voxel is associated with a sensor view candidate, which fully defines an end-effector pose. Inverse kinematics is used to check if a solution exists for such a sensor view candidate. Sensor view candidates with inverse kinematics solutions are further evaluated using the proposed view gain to determine the next-best-view to measure plane segments from $S_1^c$.

### 4.2 Representative vectors

Denote the set of current measured plane segments as $S_1' = \{s_m', m = 1, \ldots, K_1\}$ and the set of predicted correspondences from the CAD model using the proposed method as $S_1 = \{s_m, s_m \in S, m = 1, \ldots, K_1\}$. The complement set of $S_1$, $S_1^c = \{s_l, s_l \in S, s_l \notin S_1\}$ contains the planes from the CAD model that do not have matched measured planes yet. Then, the NBV problem is equal to finding sensor views that best capture the unmeasured plane segments in $S_1^c$. The view directions along which either new plane segments can be captured or the already captured plane segments can be measured again to obtain a more accurate plane model are referred to as the *representative vectors*. Defining such vectors is necessary to indicate the regions of interest in the robot work space, which facilitates the evaluation of sensor view candidates.

Denote the normal vectors corresponding to the plane segments that we intend to measure as $\mathbf{n}_t$, which are expressed in the object local frame, and the representative vectors as $\hat{\mathbf{n}}_t, t = 1, 2, \ldots, T$,

which are defined in the robot base frame. The goal is to find $\hat{\mathbf{n}}_t$ that represents $\mathbf{n}_t$ so that the planar feature corresponding to $\mathbf{n}_t$ can be measured when the sensor view is aligned with (or close to) $\hat{\mathbf{n}}_t$. Note that $T = |S_1^c| = L - K_1$ if only unmeasured plane segments are considered.

We use the following approach to compute $\hat{\mathbf{n}}_t$ using $\mathbf{n}_t$. Let $\mathbf{n}_m$ and $\mathbf{n}'_m$, respectively, be the normal vectors corresponding to plane segments $s_m \in S_1$ and $s'_m \in S'_1$. Denote $\mathbf{M}_1 = [\dots \mathbf{n}_m \dots]$ as the matrix whose columns are $\mathbf{n}_m$ and $\mathbf{M}'_1 = [\dots \mathbf{n}'_m \dots]$ is the matrix whose column vectors are $\mathbf{n}'_m$. Denote $\mathbf{M}_1^\dagger$ as the pseudoinverse of matrix $\mathbf{M}_1$, and $\mathbf{b}_m$ as the $m$-th row vector of $\mathbf{M}_1^\dagger$. Denote $\mathbf{f}_q$ and $\mathbf{f}'_q$, respectively as the basis vectors of the null space of $\mathbf{M}_1$ and $\mathbf{M}'_1$. Let $Q$ denote the dimension of $\mathbf{M}'_1$; note that $Q$ is either 1 or 2. We generate a set of representative vectors $N = \{\hat{\mathbf{n}}_t, t = 1, \dots, T\}$, using the following equation:

$$\hat{\mathbf{n}}_t = \begin{cases} \sum_{m=1}^{K_1} \alpha_m^t \mathbf{n}'_m + \sum_{q=1}^{Q} \beta_q^t \mathbf{f}'_q, \text{ if rank}(\mathbf{M}'_1) < 3 \\ \sum_{m=1}^{K_1} \alpha_m^t \mathbf{n}'_m, \text{ if rank}(\mathbf{M}'_1) = 3 \end{cases} \tag{4.1}$$

where $\alpha_m^t = \langle \mathbf{b}_m, \mathbf{n}_t \rangle$ and $\beta_q^t = \langle \mathbf{f}_q, \mathbf{n}_t \rangle$. Notice that the term $\sum_{m=1}^{K_1} \alpha_m^t \mathbf{n}'_m$ is a linear combination of the measured plane normal vectors. The coefficients of the combinations are determined by expressing the corresponding CAD model normal vectors using the measured normal vectors as bases. The term $\sum_{q=1}^{Q} \beta_q^t \mathbf{f}'_q$ contains information of the CAD model normal vectors in the null space of $\mathbf{M}'_1$. Without uncertainties, $\hat{\mathbf{n}}_t = \mathbf{R}\mathbf{n}_t$, where $\mathbf{R}$ is the rotation matrix obtained through point cloud registration. The generated representative vectors are utilized to evaluate sensor view candidates.

### 4.3 Sensor view candidates

With the defined representative vectors, which represent the directions along which the sensor can measure more planes, we proceed to define sensor view candidates, and a view gain as a criterion to select among all the possible sensor view candidate the next-best-view.

One can choose to randomly sample a set of sensor views as candidates or uniformly sample a specific region. Further, since the sensor is mounted on the robot end-effector, a sensor view

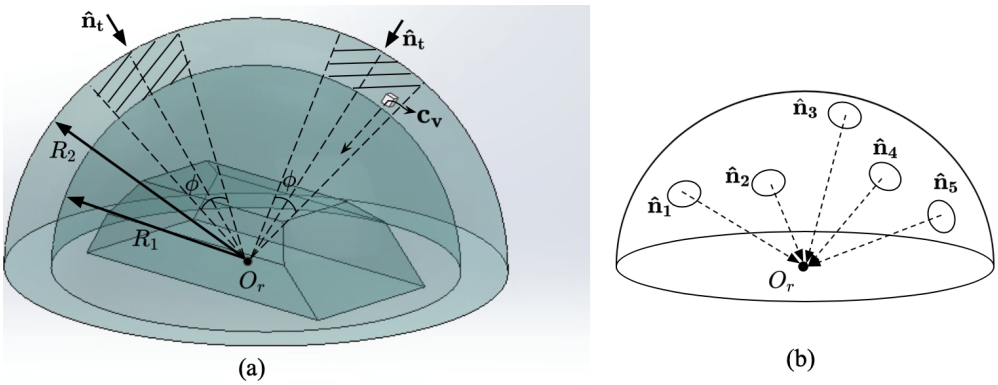



Figure 4.2: Generation and evaluation of sensor views. (a) The hatched regions between the two hemispheres with radius $R_1$ and $R_2$ are considered as the search regions for the NBVs. Each search region is discretized into many voxels, and the view gain is calculated for each voxel to determine the view with the largest view gain. The angle $\phi$ is determined by the sensor view angle constraint. The sensor view direction at a voxel (shown as white cube with center $C_v$ in the hatched region) is defined to be pointing towards the center $O_r$. (b) An example when five representative vectors are generated, five regions are evaluated to select the NBV by considering the current camera location.

candidate is only valid if an inverse kinematics solution exist such that the robot can place the sensor at the view candidate position and along the view candidate direction. The method presented here relies on the discretization of the robot workspace into voxels, and defining sensor view candidates for each voxel, followed by checking the inverse kinematics feasibility of each view candidate.

To avoid searching the entire robot workspace, which is inefficient, a search region around the workpiece is first defined. A hemispherical region of radius $R_1$ is chosen to avoid robot collision with the workpiece and also satisfy the minimal distance requirement of the sensor for taking measurements. Further, vision sensors usually can measure objects that are within a certain distance range, for which another sphere with radius $R_2$ is defined, and is concentric with the sphere with radius $R_1$. The hemispherical shell between the two hemispheres of radius $R_1$ and $R_2$ with center $O_r$ represents the region that satisfies the sensing range, as shown in Fig. 4.2(a).

Next, only regions around representative vectors are defined as regions of interest for defining view candidates, which further reduces the computation cost. Cones of angle $\phi$ with vertex at $O_r$ and axes as the representative vectors are generated. The angle $\phi$ models the view angle constraint

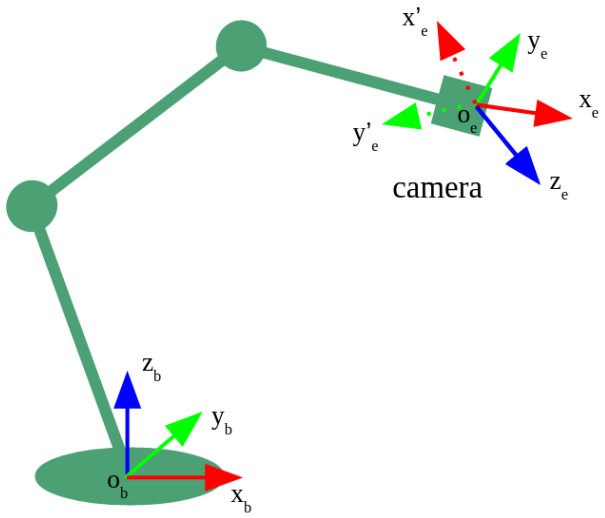



Figure 4.3: The camera has the freedom to rotate around its $z$-axis, the desired sensor view direction. The two poses of the end-effector corresponding to $o_e x_e y_e z_e$ and $o_e x'_e y'_e z'_e$ are used to calculate the inverse kinematics.

of the sensor that gives good view quality of the plane segments. The physical meaning of $\phi$ is that the sensor views that have angle smaller than $\phi$ with $\hat{\mathbf{n}}_t$ are considered to be able to measure the plane segment represented by $\hat{\mathbf{n}}_t$. For each $\hat{\mathbf{n}}_t$, we have such a cone, and the desired sensor locations to capture $\hat{\mathbf{n}}_t$ lie in the search regions, the intersection of the hemispherical shell and the cones (shown as the hatched regions in Fig. 4.2(a)).

Now, we discretize the search regions into voxels (cubes) using Octomap [96]. For each voxel, we define a sensor view candidate. The position of each sensor view candidate is the center of the voxel, and the view direction points towards $O_r$ from the voxel center. These are the sensor view candidates to be further evaluated.

Since some sensor view candidates may not be feasible for the robot, the kinematic feasibility of each sensor view candidate is checked by calculating the inverse kinematics of the robot at that location. When searching for inverse kinematics, the six degree-of-freedom (DoF) of the robot end-effector should be defined. Without loss of generality, assume the $z$-axis of the sensor frame represents the sensor view direction. Since only the $z$-axis of the sensor frame is defined, the robot end-effector has the degree-of-freedom to rotate around the $z$-axis of the sensor frame if

the sensor $z$-axis is not aligned with the last rotation axis of the robot. Thus each sensor view candidate is rotated by $18^\circ$ until a valid inverse kinematics solution is obtained or the accumulated rotation angle reaches $360^\circ$, as shown in Fig. 4.3. If a valid inverse kinematics solution is obtained, self-collision of the robot is further checked using libraries from Moveit[1].

After defining the sensor view candidates and checking the inverse kinematics feasibility and self-collision, next step is to find the best sensor view. The following view gain is proposed, which provides the balance between the travelling distances of the end-effector and how close a sensor view direction is with all the $\hat{\mathbf{n}}_t$:

$$G(\mathbf{c}_v) = \sum_{t=1}^{T} x_t \left(1 - \frac{d(\mathbf{c}_{ee}, \mathbf{c}_v)}{2R_2}\right) \langle \mathbf{c}_v - O_r, \hat{\mathbf{n}}_t \rangle \tag{4.2}$$

where $\mathbf{c}_v$ is the voxel centroid, $\mathbf{c}_{ee}$ is the current sensor location, $d(\mathbf{c}_{ee}, \mathbf{c}_v)$ is the Euclidean distance between $\mathbf{c}_{ee}$ and $\mathbf{c}_v$, $x_t$ is a binary variable that represents the view angle constraint $\phi$ of the vision sensor. $x_t = 1$ if $\langle \mathbf{c}_v - O_r, \hat{\mathbf{n}}_t \rangle \geq \cos(\phi)$, and 0 otherwise.

The proposed view gain is inversely proportional to the distance to be traveled by the end-effector from its current position to the next potential view. The closeness of the view direction with each $\hat{\mathbf{n}}_t$ is quantified by the angle between the view direction and each $\hat{\mathbf{n}}_t$, which indicates how well the sensor can capture the planes that $\hat{\mathbf{n}}_t$ represent. The view with the largest view gain is selected as the NBV. Note that $d(\mathbf{c}_{ee}, \mathbf{c}_v)/2R_2$ discounts the view gain by using the travel distance. Thus, the view gain expressed in (4.2) as the sum over all identified view directions represents to what extent the planes associated with $\hat{\mathbf{n}}_t$ can be seen at a voxel.

Figure 4.2(b) provides an example showing the five generated representative vectors $\hat{\mathbf{n}}_1$-$\hat{\mathbf{n}}_5$ when five plane segments are to be measured, five corresponding regions are discretized and evaluated considering the current camera location using the view gain. The NBV is a sensor view that can measure the most number of planes represented by $\hat{\mathbf{n}}_1$-$\hat{\mathbf{n}}_5$ while considering the traveling distance.

[1]Moveit: `https://moveit.ros.org/`

## 4.4 Simulation and experimental results

Numerical simulations and real-time experiments are conducted with a UR5 robot to evaluate the proposed strategy. The numerical simulation environment consists of the UR5 robot arm with a simulated Kinect sensor mounted on the robot flange. The Kinect sensor can measure the workspace and generate point cloud data. Software modules available in Robot Operating System (ROS), such as Gazebo and Rviz, are used for motion planning, point data processing (filtering, voxelizing, etc.) and visualization. For experiments, a 3D sensor (Ensenso N35) is mounted on the UR5 robot end-effector. A 3D workpiece (green in color) is placed on the table in the robot workspace; this workpiece has eight plane segments with one plane not visible because of resting on the workbench. The simulation environment and experimental setup are shown in Fig. 4.1. The proposed strategy is applicable with any sensors whose outputs are point clouds. The background point clouds in both numerical simulations and experiments are removed by applying a filter since the robot workspace and robot base location are known. The actual workpiece location is obtained by probing the workpiece using the robot at several locations. Some of the results are published in [36, 97].

### 4.4.1 Representative vectors

Figure 4.4 shows four cases where different number of representative vectors are generated in both simulations and experiments. Notice that the cases shown are for different individual initial views. In Fig. 4.4(a) and (b), two and five plane segments were extracted from the point clouds with blue arrows representing the normal vectors of the plane segments; representative vectors (orange) were then obtained by using the predicted plane-to-plane correspondence. The same strategy was applied in the experiments. In Fig. 4.4(c) and (d), five and four plane segments were extracted from the captured point clouds from Ensenso sensor, two and three representative vectors were generated based on the predicted correspondence by using the proposed registration method.

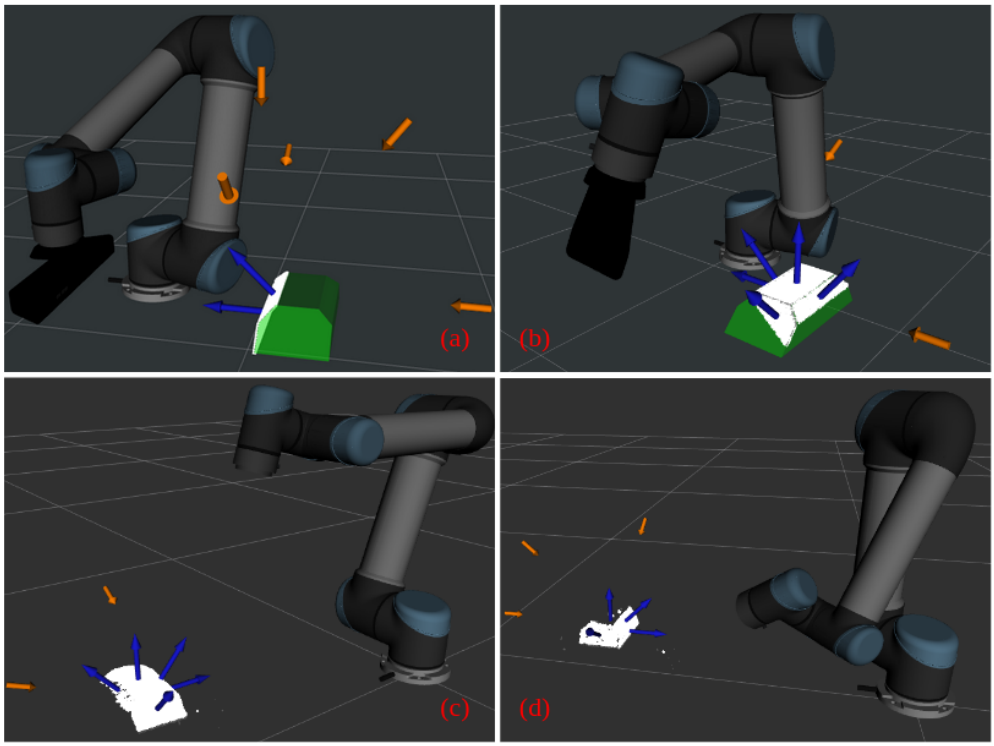



Figure 4.4: The generated representative vectors in both simulation and experiments are shown in orange arrows. For simulations shown in (a)-(b): two/five plane segments are captured by the sensor in Rviz and segmented, five/two representative vectors ($\hat{\mathbf{n}}_t$) are generated accordingly. For experiments shown in (c)-(d): point clouds from Ensenso are segmented, and two/three representative vectors are generated, respectively.

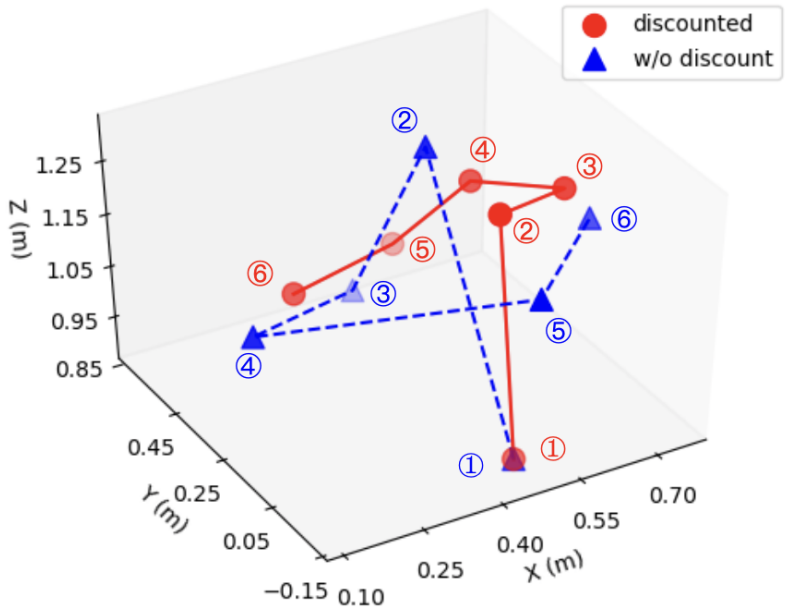



Figure 4.5: One plane segment is in the initial sensor view. Comparison of the generated six views with and without travel distance discounting factor. The initial sensor locations are the same for two scenarios. The order of the views is indicated in numbers. The total traveled Euclidean distances are 1.481 m ( with discounting factor) and 2.647 m (without discounting factor).

### 4.4.2 Effect of the travel distance discounting factor

Numerical simulation results are provided to demonstrate the effect of the travel distance discounting factor in the proposed view gain. NBVs were selected for the sensor to measure all seven visible plane segments of the workpiece.

Figures 4.5 and 4.6 present the generated view sequences (NBVs) of two cases where one plane segment is visible to the sensor initially (Fig. 4.5) and two plane segments are visible to the sensor initially (Fig. 4.6). For this simulation, one plane segment is only considered fully measured if the angle between its normal vector and the sensor is within the view angle constraint ($\phi$). As a result, there are totally six views in Fig. 4.5 and five views in Fig. 4.6. Note that in practice, plane segments may still be measured even if their normal vectors are with angles larger than ($\phi$) with the sensor view direction but the quality of the obtained point cloud may deteriorate. For each case, the views generated when travel distance discounting factor is considered are compared with

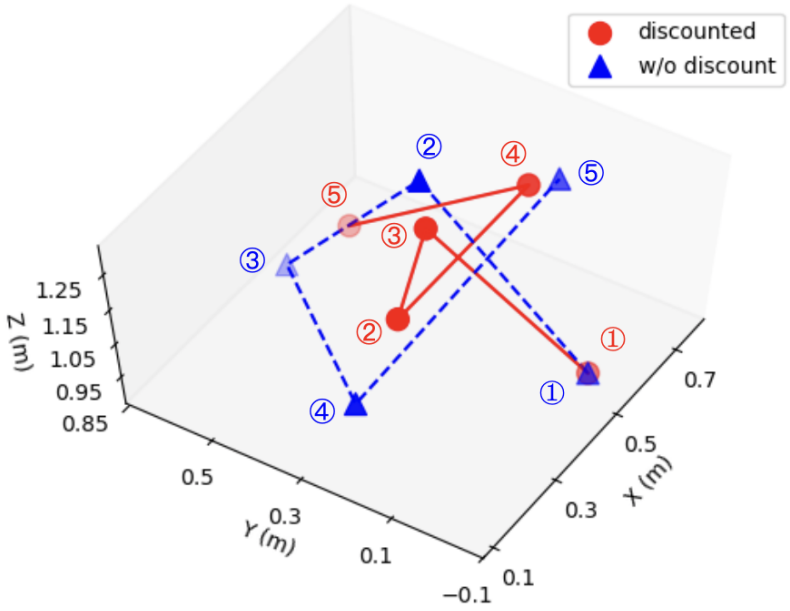



Figure 4.6: Two plane segments are in the initial sensor view. Comparison of the generated five views with and without travel distance discounting factor. The initial sensor locations are the same for two scenarios. The order of the views is indicated in numbers. The total traveled Euclidean distances are 1.436 m ( with discounting factor) and 2.266 m (without discounting factor).

the views when travel distance is not discounted. For the same number of views (six for Fig. 4.5 and five for Fig. 4.6), the view sequences are more compact and total traveling distances are much shorter (1.481 m and 2.647 m for Fig. 4.5, 1.436 m and 2.266 m for Fig. 4.6) when the travel distance discounting factor is included for both two cases, indicating the necessity of discounting the travel distance in evaluating view gain.

### 4.4.3 Next-best-view and object pose estimation

Results of work space evaluation and the obtained NBV are presented here. The effectiveness of the proposed view gain is checked by the newly measured plane segments when the sensor is moved to the NBV.

The voxels in the search regions are evaluated using Eq.(4.2). Figure 4.7 provides an example of the proposed localization process. Four plane segments (blue arrows) were obtained in Fig. 4.7(a), and three representative vectors were generated as shown in orange arrows. As

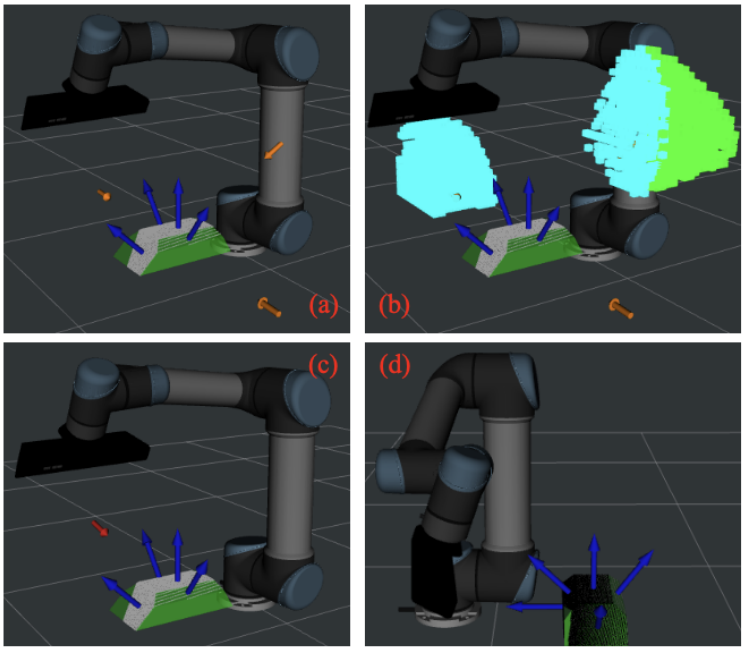



Figure 4.7: Two iterations when applying the proposed localization method: (a) At current step, three plane segments on the workpiece are not measured, three $\hat{\mathbf{n}}_i$ are generated (orange arrows); (b) regions around each $\hat{\mathbf{n}}_i$ are evaluated, one region is discarded for that no feasible inverse kinematic solutions for the robot exist; (c) the selected NBV (red arrow); (d) robot at the NBV, more plane segments are captured

Table 4.1: Workpiece Localization Results (rx,ry,rz,x,y,z) with rotation angles in radian and translation in mm (S: simulation, E: experiment)

| Data | Initial misalignment | | Localization Errors | |
|---|---|---|---|---|
| | Rotation | Translation | Rotation (·1e-3) | Translation |
| S1 | (0.2, 0.2, 0.2) | (20, 20, 20) | (0.5,1.0,0.4) | (1.18,3.43,2.33) |
| S2 | (0.3, 0.3, 0.5) | (20, 20, 20) | (1.0,0.2,2.0) | (2.23,2.7,0.47) |
| S3 | (0.2, 0.2, 0.2) | (50, 50, 50) | (1.7,0.4,0.2) | (1.17,2.19,1.71) |
| S4 | (0.3, 0.3, 0.5) | (50, 50, 50) | (1.1,0.5,2.6) | (2.69,1.27,2.52) |
| E1 | (0.2, 0.2, 0.2) | (20, 20, 20) | (9.2, 1.8, 0.7) | (6.44,0.17,0.38) |
| E2 | (0.3, 0.3, 0.5) | (20, 20, 20) | (1.2,20.4,18.8) | (0.5,2.12,0.15) |
| E3 | (0.2, 0.2, 0.2) | (50, 50, 50) | (8.4,4.2,8.3) | (7.45,1.39,1.14) |
| E4 | (0.3, 0.3, 0.5) | (50, 50, 50) | (9.8,4.6,3.7) | (1.50,3.72,4.75) |

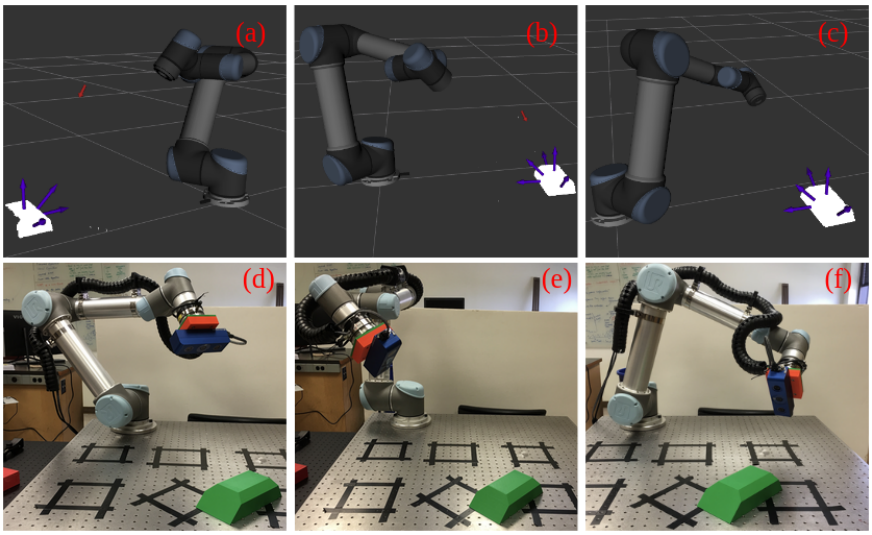



Figure 4.8: Two iterations of finding NBVs to localize the workpiece using the experiment setup: (a)-(c) point clouds and extracted plane segments visualized in Rviz, (d)-(f) physical workspace and the robotic system corresponding to (a)-(c).

shown in Fig. 4.7(b), regions around the representative vectors were evaluated using the view gain. Since one representative vector is not feasible to the robot, only two clusters of voxels are shown in Fig. 4.7(b). The visualization of the view gain of each voxel in the search region uses tools from [98]. The view gain decreases when the voxel color changes from blue to green. Thus, the obtained NBV shown in Fig. 4.7(c) (the red arrow) is selected since the voxel is closer to the current sensor location and also leads to measuring new plane segments. In Fig. 4.7(d), the robot moved to the NBV and the sensor measured a new plane segment.

The nominal workpiece locations are set to multiple different values with respect to the actual workpiece in both simulation and experiments, and a sample of results are provided in Table 4.1 (indicated by "Initial misalignment"). Figure 4.8 shows two iterations of the localization experiment.

Relatively larger translation errors are observed in the experimental results when compared to the results from numerical simulations. Several potential reasons include: (1) the errors in the extrinsic camera parameters during hand-eye calibration; (2) the errors in the robot kinematic parameters; and (3) the errors in the actual workpiece locations in the robot base frame. In particular,

robot kinematic uncertainties seems to play a larger role in these errors, and the localization accuracy also changes when the sensor views are at different locations in the robot workspace since the measured point clouds are transformed into the robot base frame and merged before segmentation.

## 4.5 Conclusions

This chapter has presented a plane-based next-best-view planning strategy for automatic object pose estimation. The goal is to estimate object pose by using vision sensing and to strategically obtain measurements. The plane-based point cloud registration method is used to generate representative vectors, which has been shown to indicate informative sensor view directions to measure more planar features on the object from both simulation and experimental results. The data collection process during pose estimation has been carried out sequentially by finding the NBVs under the proposed view gain criterion. The shorter total traveling distance, along with the measurement of additional plane segments after each NBV, indicates the efficacy of the proposed NBV strategy. Results from both numerical simulations and experiments for various object location scenarios show that the proposed strategy is effective in estimating the object pose. The following two chapters present view planning strategies that do not rely on the planar features of the objects.

# 5. POINT CLOUD ANALYSIS FOR POSE ESTIMATION

The point cloud registration and the next-best-view method for pose estimation presented in the previous two chapters require the object to have planar features. In this chapter, strategies that are agnostic to the shape of the object are developed to evaluate the collected point clouds in order to determine informative sensor view directions. Specifically, we analyze both the quantity and quality of the collected point clouds. Quantity analysis provides sensor views to collect more point clouds whereas quality analysis pinpoints locations in the current point cloud that should be measured again. We present simulation results showing the correlation between point cloud quantity/quality and registration accuracy, which motivates and guides the proposed point cloud evaluation criteria. The next chapter presents how the point cloud analysis is utilized to plan sensor views for point cloud collection*.

## 5.1 Point cloud quantity analysis

### 5.1.1 A motivation example

This section uses examples to show the correlation between the point cloud quantity and the point cloud registration accuracy. The point cloud registration accuracy is quantified by two values: the fitness of registration and the Root Mean Square Errors (RMSE) of the inlier points. The registration algorithms are state-of-the-art, including the partial-to-complete registration algorithms. The goal is to show that increasing point cloud quantity works across different registration algorithms. Thus registration results serve as motivation for generating sensor views by increasing point cloud quantity.

Figure 5.1(a) shows the 3D object and the sampled complete point cloud used for simulation. Figure 5.1(b) is one case where the initial point cloud cannot be registered well with the complete point cloud. The rotation error when expressed in axis-angle representation is 3.14 rad, translation

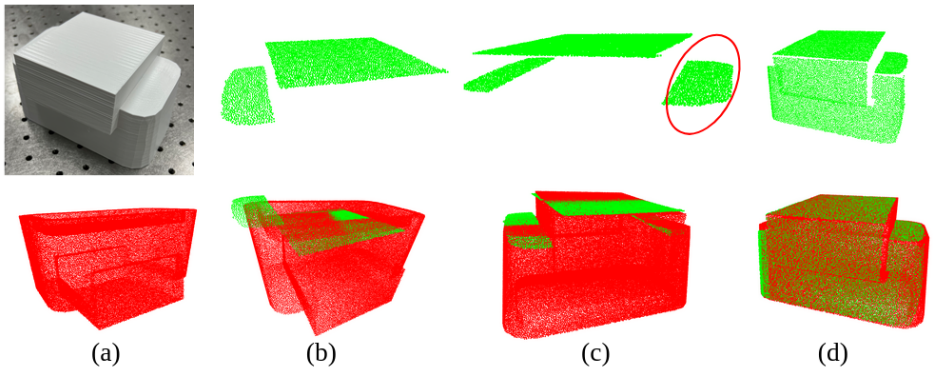



Figure 5.1: Point cloud quantity increase that improves registration accuracy.

errors are [-0.016, -0.001,-0.075] meter along the $x$-, $y$-, $z$- axis. Figure 5.1(d) shows the point cloud with increased quantity and registered point clouds with rotation error of 0.003 rad, translation errors [-1.0, -0.2,-0.1] mm, which indicates the improvement in the point cloud registration.

However, Figure 5.1(c) presents one case where the registration accuracy was not increased even though the point cloud quantity is increased. The increased part of the point cloud is highlighted in the circle. The registration has rotation error of 3.05 rad, translation errors of [-0.9, -1.0,-0.5] mm.

The fact that the increased point cloud is parallel to the rest of the point cloud can be the cause that registration is not improved, which highlights that the data collection strategy for 3D reconstruction and pose estimation should be different: while 3D reconstruction seeks to increase the coverage of the entire object, point cloud that helps registration is more important for object pose estimation. To this end, we introduce a criterion for point cloud quantity evaluation. The sensor views to increase the value of the criterion is expected to increase registration accuracy.

### 5.1.2 Point cloud quantity evaluation

*Angular spread* was first mentioned in[90]. The angular spread of a tessellated cell of a unit sphere is equal to the angle of the cone formed by the cell when connected to the center of the sphere. For a point cloud $P$, the normal vectors can be calculated using the method introduced in Chapter 2. In this work, angular spread is extended to be used with a point cloud. Denote the set

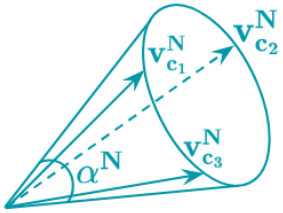



Figure 5.2: Given the three normal vectors $\mathbf{v}^N_{c_1}, \mathbf{v}^N_{c_2}, \mathbf{v}^N_{c_3}$ that represent the three cluster centers of the normal vectors set $N$ of a point cloud. The angular spread is defined as the angle of the cone, which has $\mathbf{v}^N_{c_1}, \mathbf{v}^N_{c_2}, \mathbf{v}^N_{c_3}$ on the cone base

of normal vectors $N$ is grouped into three clusters $N = \{N_1, N_2, N_3\}$ by applying the spherical K-means algorithm[1] [79]. The centers of the three clusters are given by the unit vectors $\mathbf{v}^N_{c_1}, \mathbf{v}^N_{c_2}, \mathbf{v}^N_{c_3}$ and the weights of the clusters (ratio of number of normal vectors in a cluster to the total number of normal vectors) are given by $w_{c_1} = \frac{|N_1|}{|N|}$, $w_{c_2} = \frac{|N_2|}{|N|}$, and $w_{c_3} = \frac{|N_3|}{|N|}$. We define the angular spread of $N$, $\alpha_N$, as angle of the cone that has the unit sphere center as the apex, and the base defined by $\mathbf{v}^N_{c_1}, \mathbf{v}^N_{c_2}, \mathbf{v}^N_{c_3}$, as illustrated in Fig. 5.2.

*Remark:* The goal of the quantity analysis is to determine how well the collected point clouds can constrain the object. Since three vectors are adequate to span the 3D space, one can choose to define a matrix that has the normal vectors of a point cloud as row vectors and use the rank and singular values of the matrix as indicators of the point cloud quantity. However, the calculation of the matrix rank is prone to errors in the normal vectors. Instead, we study the distribution of normal vectors of a point cloud by clustering the normal vectors into three clusters. The defined angular spread uses the three cluster centers as an indicator of the point cloud quantity, which is less affected by noisy data. Thus $\alpha_N$ can be considered as the geometric counterpart of the rank of the matrix formed by all $\mathbf{n}_i$: it quantifies the spread of $\mathbf{n}_i$. When $\mathbf{v}^N_{c_1}, \mathbf{v}^N_{c_2}, \mathbf{v}^N_{c_3}$ are equal, $\alpha_N = 0$, corresponding to the matrix having rank of 1.

Another way of quantifying the point cloud quantity is to study the tetrahedron formed by the unit sphere center $o$ and the three cluster centers: $o\mathbf{v}^N_{c_1}\mathbf{v}^N_{c_2}\mathbf{v}^N_{c_3}$. The volume of $o\mathbf{v}^N_{c_1}\mathbf{v}^N_{c_2}\mathbf{v}^N_{c_3}$ can serve as a measure of the point cloud quantity.

[1]Implementation of the algorithm can be found at: https://github.com/jasonlaska/spherecluster

## 5.2 Point cloud quality analysis

Another factor that affects the point cloud registration accuracy is the quality of the point clouds. The quality of a point cloud can be interpreted as the noise level in the point cloud or the percentage of points that represent the surfaces from which they are captured. Point cloud with low noise level is considered to have high quality. Thus, the emphasis of quality evaluation is to find out the locations in the point cloud that require additional measurements to improve the quality and potentially improve the registration accuracy. We first present the correlation between point cloud quality and registration accuracy to both motivate and lay the foundation for the development of the method. The following are some important definitions:

- inliers: points that reflect the surfaces from which they are sampled
- outliers: points that do not reflect the surfaces from which they are sampled
- point cloud quality: degree to which the points reflect the object geometries, or the number of inliers
- problematic areas: areas in the point cloud that have poor qualities

### 5.2.1 A motivation example

One example is shown in Fig. 5.3. The results are from an Ensenso stereo camera N35 mounted on the flange of a UR5 robot. Point clouds from physical sensors are used instead of adding artificial noises to reflect the realistic irregularities in practice. Figure 5.3 shows examples of how the obtained point cloud quality is affected by the sensor view and sensor location. The point cloud on the left is the initial point cloud. One surface is not complete and has disconnected holes. As the sensor moved to another location and the sensor view changed, the incomplete surface had dense point clouds with fewer outliers. But another surface was captured with poor quality, which was improved by moving the sensor to the third location. The obtained point cloud is shown in the right figure. This example shows how the point cloud quality can be improved by continuously changing the vision sensor.

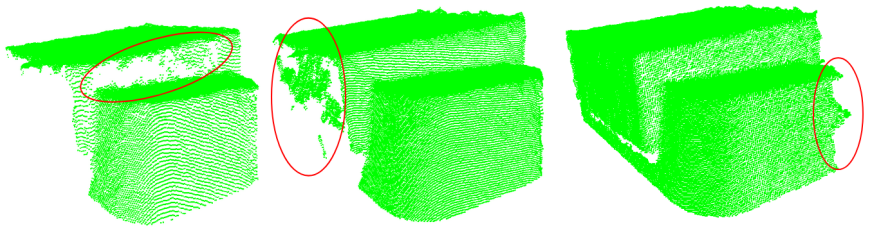


Figure 5.3: Point cloud quality changes as the vision sensor moves to different view angle and view location(left to right). Irregularities in the point clouds are highlighted by red circles.

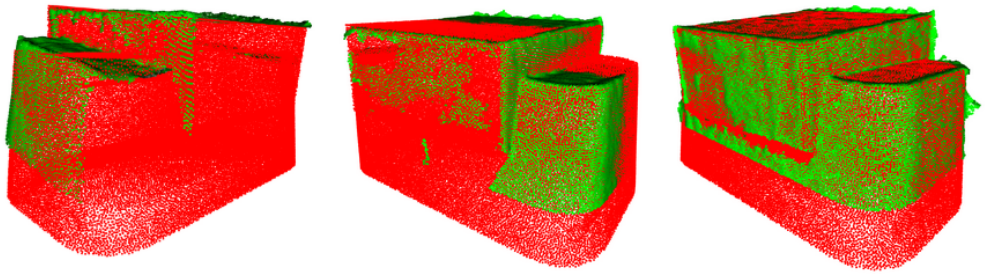


Figure 5.4: Examples of how point cloud quality affects the registration accuracy.

As can also be seen from Fig. 5.3, the quality of the point cloud can be affected by the holes on the surface and distortions on the edges. Both cases of quality issues can be effectively mitigated by changing the sensor locations and view directions.

Figure 5.4 shows the registration results of the above three point clouds by using Go-ICP. Two criteria are used to evaluate the registration accuracy since the ground truth location of the object are not known: fitness and the RMSE. Fitness represents how many points in the measured point cloud is matched with a correspondence in the complete point cloud whereas inlier RMSE indicates the average Euclidean distances between the point correspondences. The fitness and inlier RMSE for the registration from the left to the right are: 0.904 and 1.2, 0.918 and 1.2, 0.994 and 0.7, indicating that point cloud quality affects the accuracy of the registration.

Now the task is to identify the locations in the point cloud that need to be measured again, i.e., the problematic locations in the point cloud, and how to generate sensor views in order to measure the locations to reduce the outliers and increase registration accuracy.

*Remark:* Causes of the quality issues in the point clouds can come from various sources, such as the placement of the vision sensor, light conditions, object surface texture, reflections, etc. This work analyzes how the placement of the vision sensor affects the point cloud quality and how to improve the quality by changing the vision sensor location, which is discussed in the next chapter.

### 5.2.2 Point cloud quality evaluation

We first define the criteria that are used to find the problematic areas for a point cloud. The proposed criteria can also be used to filter the outliers and generate sensor views to measure the problematic areas.

Given a point cloud $P = \{\mathbf{p}_i, i = 1, \ldots, \mathbf{p}_i \in \mathbb{R}^3\}$, which is collected using a vision sensor under sensor view direction $\mathbf{v}_s, \mathbf{v}_s \in \mathbb{R}^3$. Denote the set consisting of the normal vectors corresponding to the point cloud by $N = \{\mathbf{n}_i, i = 1, \ldots\}$. Note that $P, N, \mathbf{v}_s$ are defined in the same coordinate frame. We develop a set of quality criteria to evaluate the point cloud quality and label each point as either *inlier*, points that meet this quality criteria are saved for pose estimation, or *outlier*, points that do not meet this criteria. Denote the set of outlier points as $P_b$. The positions and normal vectors of outlier points are used to find sensor views such that outlier points can be measured again to improve the point cloud quality.

Incidence angle is defined as the angle between the sensor view and the surface normal direction. Incidence angle has been used in reconstruction to evaluate voxel quality [65], determine scanning direction [66], and remove mixed or discontinuous pixels [67]. In this work, dynamic incident angle intervals are used to identify outliers in the point clouds.

We consider two independent criteria to evaluate the quality of every point in the point cloud. (1) View angle criterion: the incidence angle $\theta_i$, defined as $\theta_i = \angle(-\mathbf{v}_s, \mathbf{n}_i)$, lies in a preferred interval $[\theta_l, \theta_u]$ within which the obtained point cloud is expected to closely represent the measured surface. We propose to employ a different interval for each point cloud that can better identify outliers. Denote the minimum and maximum angle between $\mathbf{v}_{c_1}^N, \mathbf{v}_{c_2}^N, \mathbf{v}_{c_3}^N$ and $\mathbf{v}_s$ as $\theta_l^v$ and $\theta_u^v$. $\delta_c$ is a small constant. Denote $\theta_l' = \max\{\theta_l, \theta_l^v - \delta_c\}, \theta_u' = \min\{\theta_u^v + \delta_c, \theta_u\}$. The interval $[\theta_l', \theta_u']$ can be chosen dynamically for each measured point cloud. (2) Statistical criterion [89]: the average

Euclidean distance of a point to its neighbors, $\bar{d}(\mathbf{p}_i, P_k(\mathbf{p}_i))$, is less than a value, where $P_k(\mathbf{p}_i)$ is the set of k-nearest-neighbors of $\mathbf{p}_i$, $P_k \subseteq P$. The view angle criterion captures the preferred sensing cone of a given vision sensor for better sensing quality. Points that have larger average distance from its neighbors than the average distance across the entire point cloud are discarded for the current pose measurement when using the statistical criterion; however, the information from these points and their normals are utilized to obtain additional sensor views. Thus, the set of inlier points is defined as:

$$P_g = \{\mathbf{p}_i | \mathbf{p}_i \in P, \theta_i \in [\theta_l, \theta_u], \bar{d}(\mathbf{p}_i, P_k(\mathbf{p}_i)) \leq d_0\} \tag{5.1}$$

where $d_0$ is the average distance between a point and its neighbor in the entire point cloud. The set of outlier points is given by $P_b = P \setminus P_g$. While not used for point cloud registration, $P_b$ contains important information to determine additional sensor views to measure the problematic areas to improve the point cloud quality. In order to determine the sensor views, we need to determine the target position and orientation for the sensor to measure. For the target position, points in $P_b$ are clustered based on the density of points by using DBSCAN [99]. Denote $s_e, e = 1, \ldots,$ as the obtained clusters and the set of all clusters to be $S_b = \{s_e, e = 1, \cdots\}$; note that $P_b = \cup s_e$. Then we use the cluster center and cluster normal to generate sensor views instead of using points.

The reasoning behind employing the above two criteria for point cloud evaluations is as follows. When using the statistical criterion, inliers might be identified as outliers if the neighborhood size and variance parameters are not set properly in the algorithm. When using the view angle criterion, inlier points on the edges tend to be identified as outliers if normals are not calculated accurately. The common outliers resulting from both criteria has provided a better estimate of the outliers, which was corroborated experimentally.

### 5.3 Experiment results

This section presents the results of point cloud evaluation in terms of both quality and quantity. For this evaluation, the sensor was manually placed in different views to collect multiple point

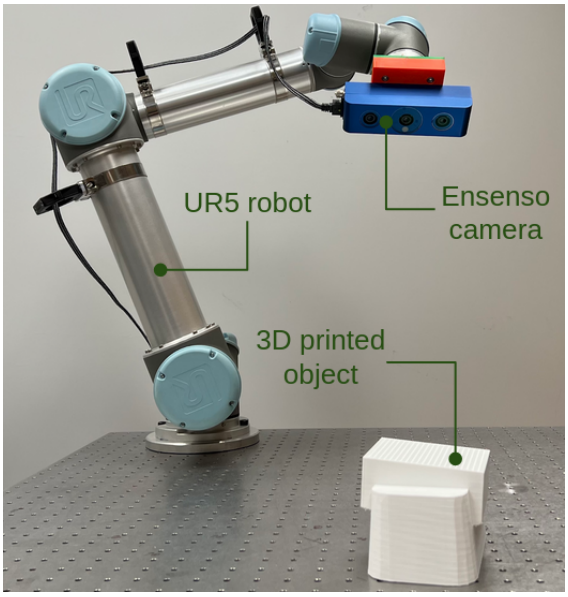



Figure 5.5: The robotic system utilized to collect point clouds

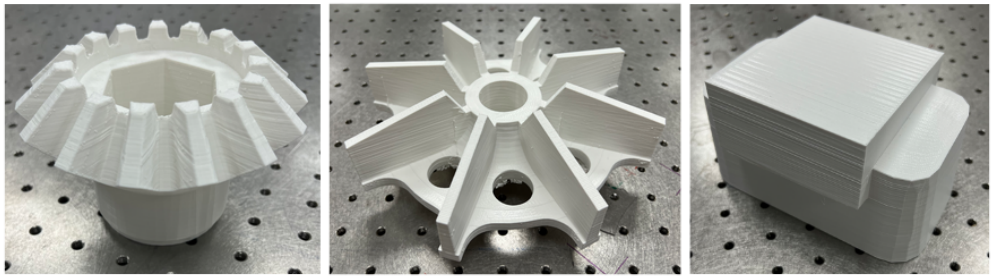


Figure 5.6: 3D printed objects used in the experiments, from left to right: gear, impeller, and a mechanical component.

clouds that can represent cases in which the quality and quantity evaluations are effective, i.e., identify outliers and check if the point cloud is adequate for pose estimation. The robotic system used in the experiments is shown in Fig. 5.5, which consists of a UR5 robot arm, an Ensenso camera, and a 3D printed object. The host computer runs Ubuntu 18. Both the camera and the robot are connected to the computer through Ethernet. The robot is controlled by the host computer using the Robot Operating System (ROS) Melodic. Three different objects are used in the experiments

(see Fig. 5.6). These object models are obtained from Fusion360 dataset[2] and MCB dataset[3]. Some of the results are published in [100].

### 5.3.1 Point cloud quantity analysis

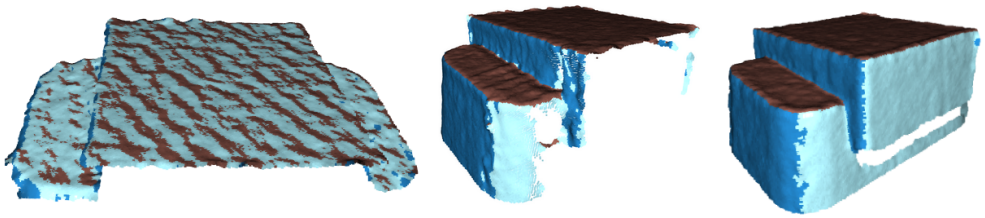
Examples are provided to show that the increase of angular spread is a good prediction for point cloud registration accuracy. Thus, it is reasonable to generate sensor views that increase the angular spread of the collected point cloud.

Fig. 5.7 shows three point clouds collected from different sensor views. The point clouds are clustered based on normal vectors with clusters highlighted in different colors. Figure 5.7(a) indicates the point cloud is not enough to fully constrain the object for pose determination. Figure 5.7(b) indicates that the point cloud quantity is theoretically adequate to estimate the object pose, but some object areas are not well-defined along certain directions (normal vectors corresponding to the light green colored points). In Fig. 5.7(c), points are distributed relatively evenly and cover the entire object, which will lead to better pose estimation. These experimental results indicate that the angular spread is a good indicator for evaluating the quantity of the point cloud and for predicting how well the point cloud can be used to estimate the object pose. The registration results for the above three point clouds are shown in Fig.5.8. The fitness and inlier RMSE are used to evaluate the registration accuracy. The fitness and inlier RMSE for the point clouds from the left to the right are: 0.887 and 1.2, 0.928 and 1.1, 0.954 and 0.59.

Another examples of three point clouds of different quantity are shown in Fig. 5.9. The angular spread for the three point clouds are (from left to right): 70.3, 71.7, and 73.1 degrees. The fitness and inlier RMSE are: 0.932 and 1.3, 0.999 and 0.6, 0.999 and 0.5. The results show that the increase of the angular spread indicates (or predicts) the improvement in registration even though the changes in the angular spread is not as significant when compared to the example in Fig. 5.9. The changes in the angular spread depends on the geometry of the object and the initial point cloud.

The top row of Fig. 5.10 shows two point clouds of a gear collected at two different sensor

[2]Fusion360: https://github.com/AutodeskAILab/Fusion360GalleryDataset
[3]MCB: https://mechanical-components.herokuapp.com



(a) (b) (c)

Figure 5.7: Point cloud quantity evaluation for cases where one (a), two (b), or three (c) distinct planes exist in the point clouds. The three clusters obtained based on the normal vectors are highlighted in three different colors and the angular spreads (in degrees) are: (a) 39.75, (b) 87.26, (c) 99.92.

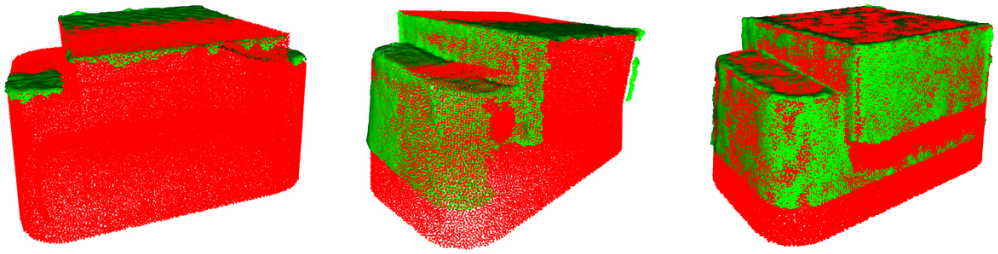


Figure 5.8: Registration results of the three point clouds shown in Fig. 5.7 by using Go-ICP.

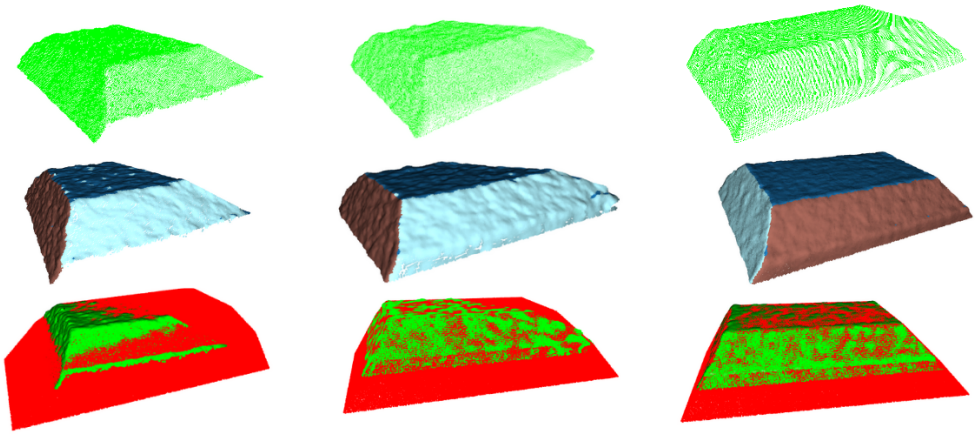


Figure 5.9: Registration results of the three point clouds of a convex object using Go-ICP. Top row: point clouds. Middle row: clustered point clouds in different colors. Bottom row: registration results, the complete point cloud is shown in red.

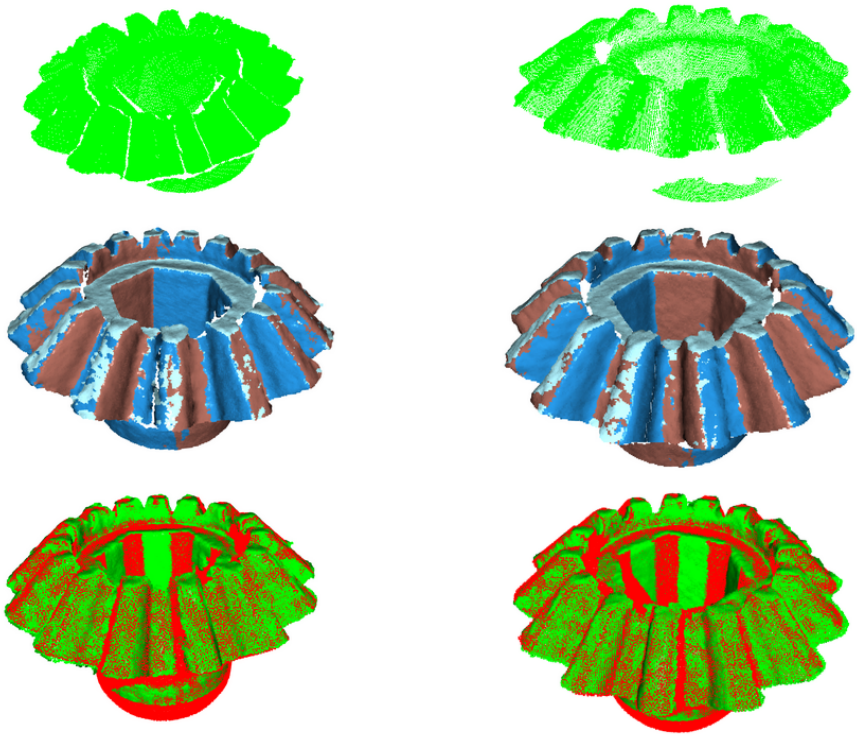


Figure 5.10: Point clouds of a gear captured at different sensor view locations. From top to bottom: point clouds, clustered point clouds, and registration results.

locations. The calculated angular spreads are $74.4°$ (left) and $74.8°$ (right). The registration fitness and RMSE are: 0.917 and 0.9, 0.913 and 0.9, respectively. This example shows that the similarities in the point clouds collected from symmetric objects leads to the low variance in the angular spreads of the point clouds, for which reason, the point cloud quality analysis is an important addition to the quantity analysis.

### 5.3.2 Point cloud quality analysis

The effectiveness of the proposed quality evaluation method is tested by using point clouds collected using vision sensor to verify the effectiveness of the proposed method in practice.

Figure 5.11 shows the point clouds collected from one sensor view and the quality evaluation by using the statistical criterion and the view angle criterion. The outlier points under each criterion is highlighted in red, and the common set of points using the two criteria is shown in (c).The

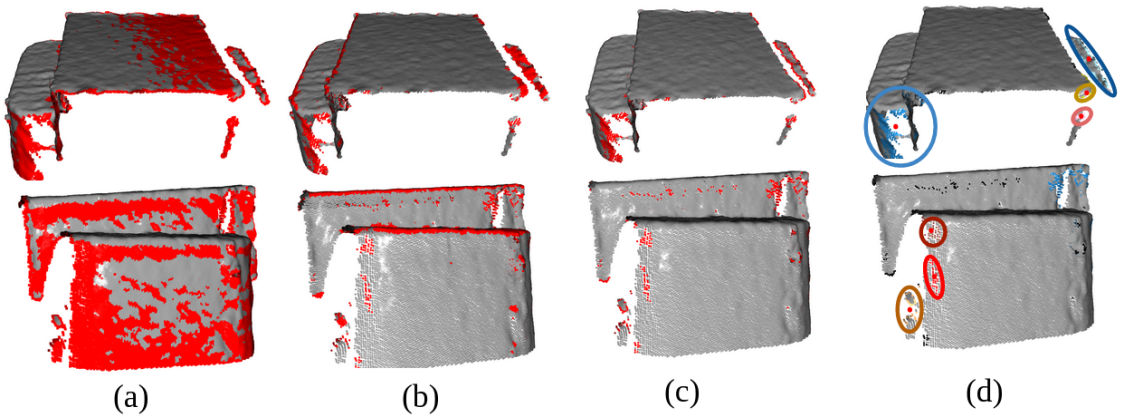



Figure 5.11: Point cloud quality evaluation: (a) outliers detected by the statistical criterion are highlighted in red, the neighbor points number and standard deviation ratio are set to 20 and 0.1 in the algorithm, (b) outliers detected using the view angle criterion are highlighted in red, (c) the common outliers of the two criteria, (d) seven clusters obtained using DBSCAN and shown using circles with geometric centers indicated using red dots.

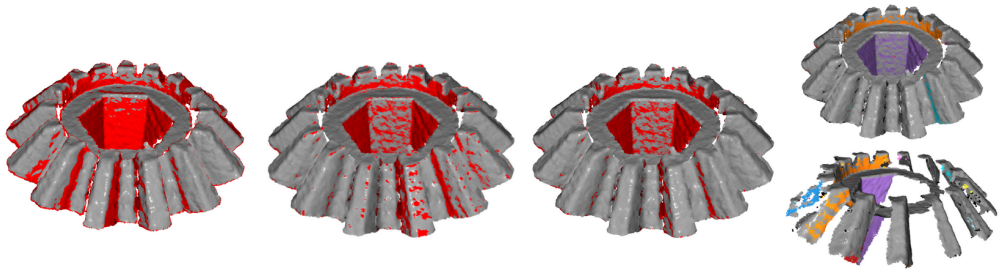


Figure 5.12: Quality evaluation of the point cloud of a gear

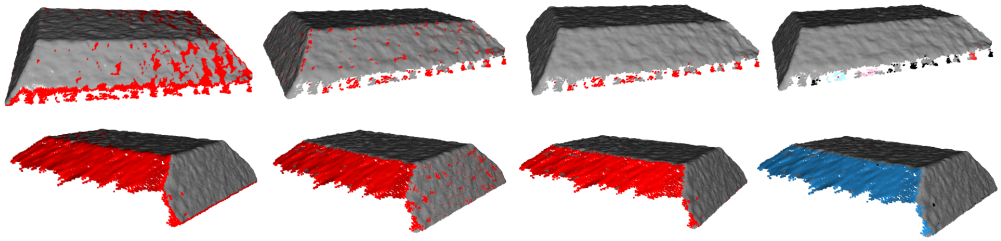


Figure 5.13: Quality evaluation of the point cloud of a convex object

clusters obtained using DBSCAN are highlighted in (d) in different colors. The statistical criterion detects large areas with significant number of inliers, while the view angle criterion identifies most edges as outlier points which is due to the inaccurate normal vectors on sharp edges when using a neighborhood of points for calculation. The combination of the two criteria identify holes and irregularities in the point clouds while reducing false positive outliers, as shown in Fig. 5.11(d).

Figure 5.12 and Fig. 5.13 present two other examples of point cloud quality evaluation. The outliers identified are highlighted in red. From the left to the right are the results of: statistical outlier criterion, view angle criterion, common point set of the two, clustered outliers in different colors. The common set of outliers can reflect the areas on the point clouds that are not measured under the sensor's preferred view angle range The gear has more identified outliers comparing to the convex object.

## 5.4 Conclusions

This chapter presents methods to analyze the collected point clouds in terms of quality and quantity, which will be utilized to generate informative sensor views for view planning. The effectiveness of the quantity criterion, angular spread, is shown through the increased registration results (fitness and RMSE) when using the point clouds of different quantities collected using the vision sensor. The proposed quality analysis criteria can identify the areas in the point clouds that have irregularities across different objects. It is observed that the object geometry and the initial point cloud affect to what degree the registration results can be improved by increasing the point cloud quantity or quality. The relative effectiveness of quantity and quality analysis may vary. Point clouds collected from objects with symmetric features usually can constrain the six DoF of the object, and objects with small surface curvatures may have fewer irregularities, which indicates the proposed quantity and quality analysis is complementary.

# 6. VIEW PLANNING BASED ON POINT CLOUD ANALYSIS

This chapter presents the strategies to generate and plan sensor views based on the point cloud analysis to improve the point cloud quality and quantity. The generation of sensor view candidates is formulated as optimization problems instead of the uniform sampling strategy used in Chapter 4. Robot configurations corresponding to the sensor views are solved by formulating a constrained nonlinear optimization problem, which reconciles the potential conflicts between the constraints from the robot kinematics and the sensing requirements. Experimental results are provided to demonstrate the effectiveness of each component. Two other view planning strategies are compared with the proposed method in terms of the effectiveness of improving estimated object poses*.

## 6.1 Sensor views from quantity analysis

For a given point cloud and its angular spread, the task is to find the sensor views that increase the angular spread such that the registration accuracy can be increased. We used the three dominant normal vectors (the centers of the three normal vector clusters) to calculate the angular spread in Chapter 5. The strategy to increase the angular spread and collect point clouds that are in favor of registration is to increase the number of points from the surfaces that the measured normal vector with the least weight represent ($\mathbf{v}_{\mathbf{p1}}^{\mathbf{N}}$). The proposed method is to first find the resultant direction of the collected normal vectors and the least weighted normal vector, then define a sensor view direction based on the two vectors.

The goal of the quantity analysis is to obtain a sensor view direction(s), $\mathbf{v}_t$, that provides a larger angular spread $\alpha_N$ than the current point cloud. Without loss of generality, assume $w_{c_1} \leq w_{c_2} \leq w_{c_3}$. We propose the following strategy to increase $\alpha_N$. Let $\mathbf{n}_a = w_{c_1}\mathbf{v}_{c_1}^N + w_{c_2}\mathbf{v}_{c_2}^N + w_{c_3}\mathbf{v}_{c_3}^N$. Let $R$ denote the plane normal to $\mathbf{n}_a$, $\mathbf{v}_{p_1}^N$ denote the projection of $\mathbf{v}_{c_1}^N$ on $R$, which is further

*©2022 IEEE. Part of this chapter is reprinted with permission from Jie Hu, Prabhakar Pagilla, "View planning for object pose estimation using point clouds: an active robot perception approach", IEEE Robotics and Automation Letters, July, 2022.

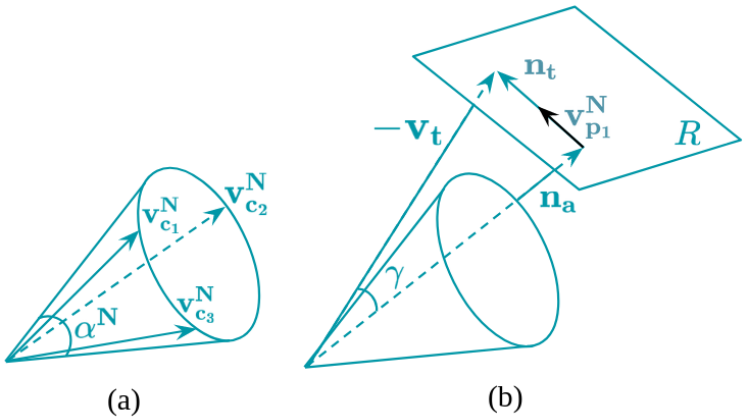



Figure 6.1: Given the three normal vectors $\mathbf{v}_{c_1}^N, \mathbf{v}_{c_2}^N, \mathbf{v}_{c_3}^N$ that represent the three cluster centers of the normal vectors set $N$ of a point cloud, (a) the angular spread; (b) shows the proposed view direction $\mathbf{v}_t$ to increase the angular spread

normalized to unit length, and $\gamma$ is a scalar such that $\mathbf{n}_t = (\|\mathbf{n}_a\|\tan\gamma)\mathbf{v}_{p_1}^N$. The sensor view to increase the angular spread is selected as $\mathbf{v}_t = -(\mathbf{n}_a + \mathbf{n}_t)$, which is illustrated in Fig. 6.1(b).

The physical explanation of the above defined NBV for increasing the angular spread of a point cloud is to represent the current normal vectors using the resultant vector, then deviate $\gamma$ angle from the resultant vector along the direction that has the smallest weight, to measure surfaces with normal vectors in the neighborhood of the direction with the smallest weight. Notice that sensor views to increase point cloud quantity by using the other two vectors $\mathbf{v}_{c_2}$ and $\mathbf{v}_{c_3}$ and their projections on $R$ ($\mathbf{v}_{p_2}^N$, $\mathbf{v}_{p_3}^N$) can be defined in a similar manner:

$$\mathbf{v}_{t_2} = -(\mathbf{n}_a + \|\mathbf{n}_a\|\tan\gamma)\mathbf{v}_{p_2}^N) \tag{6.1}$$

$$\mathbf{v}_{t_3} = -(\mathbf{n}_a + \|\mathbf{n}_a\|\tan\gamma)\mathbf{v}_{p_3}^N) \tag{6.2}$$

The three sensor views can be sorted in an ascending order based on the weights of the cluster centers such that the sensor views corresponding to clusters with less weights will have higher priorities.

## 6.2 Sensor views from quality analysis

In this section, we develop a method to determine a set of sensor views to measure the areas on the object with identified outliers to collect new point clouds with higher quality. The goal is to obtain point clouds of the problematic areas with fewer outliers for better registration results. The task of finding sensor views to measure a given set of clusters of outliers $S_b$ is decomposed into two subtasks: (1) find the minimum number of sensor views that can measure $S_b$, (2) find the values of the sensor views. For (1), the problem is formulated as a minimum set cover problem, the solution of which gives which clusters in $S_b$ can be measured from one sensor view. For (2), the problem is formulated as a mixed-integer problem, the solution of which gives the values of the sensor views. We first introduce the generation of subsets and the formulation of a minimum set cover problem, then present the mixed-integer problem formulation.

### 6.2.1 A minimum number of sensor views

While the maximum number of views is equal to the number of clusters in $S_b$, i.e., measuring each cluster with one different sensor view, we are interested in determining the minimum number of sensor views since multiple clusters may be measured simultaneously. In order for some clusters to be measured with good quality simultaneously, we first define two necessary conditions that a combination of clusters have to satisfy. The first condition corresponds to respecting an incidence angle constraint; specifically, clusters are expected to be measured within a smaller incidence angle interval $[\theta_l^s, \theta_u^s]$ since we are interested in specific regions on the object. The second condition is that the distance between any clusters is within a threshold; this ensures that clusters that are beyond a certain threshold distance are not included in the same view. Notice that the incidence angle interval defined here is different from the view angle interval $[\theta_l, \theta_u]$ which is determined by the hardware, while $[\theta_l^s, \theta_u^s]$ is a user-specified interval decided based on the geometry complexity, size and other conditions. We choose $\theta_l^s$ and $\theta_u^s$ such that $\theta_l^s \geq \theta_l, \theta_u^s \leq \theta_u$. Even though the two conditions do not guarantee that the clusters will be measured with good quality, they are necessary but not sufficient conditions.

Formally, let $V_l$ denote a set of sensor views to measure the clusters in $S_b$. We aim to find $V_l^*$ such that the size of $V_l$ is minimized, and each $\mathbf{v}_l^* \in V_l^*$ is optimized with respect to the clusters to be measured. First denote the set of all possible combinations of clusters $s_e \in S_b$ as $C = \{C_f, f = 1, \ldots\}$, where each $C_f$ is a combination of some $s_e$. For a $C_f \subseteq C$, the maximum angle between any two cluster normals in $C_f$ is denoted as $\theta_{max}(C_f)$, the maximum Euclidean distance among the clusters in $C_f$ is denoted as $d_{max}(C_f)$. Define $C_v$ as the set of all combinations: $C_v = \{C_f | C_f \in C, \theta_{max}(C_f) \leq (\beta_l - \delta_l), d_{max}(C_f) \leq d_c\}$, where $d_c$ is the allowed maximum distance among the clusters. Notice that the generation of the subsets can be tuned based on the application requirements, object geometries, etc., and is the key to the minimum set cover problem since the sensor views directly depend on the clusters to be measured. Thus, the cover is minimum with respect to the two necessary conditions to be satisfied by the clusters.

The minimum set cover problem is the following: given the subsets $C_f \in C_v$, and the universe $S_b$. Find the minimum set cover, $\mathcal{C}^*$. The solution gives the minimum number of views to cover $C_v$, or equivalently, all points in $P_b$, and all clusters in $S_b$.

### 6.2.2 The optimal sensor views

Given the minimum number of views and the sets of clusters for each view to measure, we proceed to find the optimal sensor view values to measure the corresponding set of clusters. Views are optimal in the sense that all clusters are to be measured under the preferred incidence angle range $[\theta_l, \theta_u]$. We formulate and solve mixed-integer problems to find such sensor views to measure the subsets in $\mathcal{C}^*$, which is described below.

Denote the set of clusters that needs to be covered by sensor view $\mathbf{v}_a$ as $C_a$, $C_a \in \mathcal{C}^*$. Denote the normal vectors of the clusters in $C_a$ as $\{\mathbf{n}_{a_j}, j = 1, \ldots, |C_a|\}$, and the geometric center of $C_a$ as $\mathbf{p}_a$. The weight of each cluster in $C_a$ is $w_{a_j}$, the ratio of the number of points in the cluster to the total number of points in the point cloud $P$. Let $x_{a_j}$ be a binary variable, $x_{a_j} = 1$ indicates the cluster corresponding to $\mathbf{n}_{a_j}$ is viewed within the preferred incidence angle interval, and $x_{a_j} = 0$ otherwise. Then, in order to determine the view to measure all clusters in $C_a$ under the preferred

angle interval, the following problem is formulated:

$$\mathbf{v}_a^* = \max_{\mathbf{v}_a \in \mathbb{R}^3} \sum_{j=1}^{|C_a|} x_{a_j} w_{a_j} \tag{6.3}$$

subject to the following constraints:

$$\mathbf{v}_a^T \mathbf{v}_a = 1 \tag{6.4}$$

$$x_{a_j}\delta_l \leq x_{a_j} \angle(-\mathbf{v}_a, \mathbf{n}_{a_j}) \leq \beta_l, \ \ \forall j \tag{6.5}$$

The constraint in (6.5) can be written as:

$$\begin{aligned} 1 + \frac{x_{a_j}}{\cos\beta_l} \mathbf{n}_{a_j} \cdot \mathbf{v}_a \leq 0, \ \ \forall j \\ 1 + \frac{x_{a_j}}{\cos(x_{a_j}\delta_l)} \mathbf{n}_{a_j} \cdot \mathbf{v}_a \geq 0, \ \ \forall j. \end{aligned} \tag{6.6}$$

The view angle constraints that were used to determine the combinations in $C_v$ ensure the existence of a solution to the above optimization problem, which will provide a view that can measure all the clusters in $C_a$ within the preferred view angle interval. Subsequently, all pairs of $(\mathbf{p}_a, \mathbf{v}_a^*)$ are used to calculate corresponding robot poses.

The constraints in equation (6.6) contain the multiplication of integer variable $x_{a_j}$ and continuous variables $\mathbf{v}_a$. We can consider two techniques to solve the optimization problem with such constraints. The first one is to use solvers that gives the global optimal solutions, such as Alpine[1] [101]. The second technique is to relax the term $(x_{a_j}\mathbf{v}_a)$ using McCormick envelope [92], and decompose the problem into mixed-integer linear programs and non-linear programs. An implementation can be found in Pyomo[2] [102, 103].

[1] Alpine: https://github.com/lanl-ansi/Alpine.jl
[2] Pyomo: http://www.pyomo.org/

## 6.3 Robot pose determination corresponding to sensor views

In Chapter 4, sensor view candidates are generated based on the discretized search regions around the object, and followed by checking the inverse kinematics feasibility and collision. The approach presented here is to formulate a nonlinear optimization problem while considering all the constraints of the system. The advantages are: (1) avoiding explicitly defining end-effector poses, (2) avoiding discretizing and evaluating the robot work space, which are inefficient.

The constraints to be considered in the problem formulation include: (1) self-collision of the robotic system and self-occlusion caused by the robot being partially visible to the sensor; (2) visibility constraints: the object should remain in the sensor view when the sensor is at the new view; (3) minimum and maximum distance between the sensor and the object, which is intended to satisfy the field-of-distance constraint of the sensor and for safety. Denote the set of potential views to increase the point cloud quality and quantity as $V = V_l^* \cup V_t$. Let a robot pose be given by its joint coordinate vector $\mathbf{q}$, where $\mathbf{q} \in \mathbb{R}^n$ for a robot with $n$ joints. Denote the forward kinematics of the robot arm as $\mathbf{T}(\mathbf{q}) \in$ SE(3). Without loss of generality, assume that the robot end-effector $z$-axis is aligned with the sensor view direction, which is denoted as $\mathbf{v}_{ee}$ and is given by the first three elements of the third column of $\mathbf{T}$. Denote the sensor position as $\mathbf{t}_{ee} \in \mathbb{R}^3$ which is given by the last column of $\mathbf{T}$. The current robot pose is $\mathbf{q}_0$. Each of the constraints are discussed in detail.

### 6.3.1 View angle and joint angle constraints

Denote the maximum allowable angle between $\mathbf{v}_{ee}$ and $\mathbf{v}$ as $\eta$, $\mathbf{v}_{ee} \cdot \mathbf{v} \geq \cos\eta$. Let $h_1 = \cos\eta - \mathbf{v}_{ee} \cdot \mathbf{v}$. Let the $r$-th joint have a specified range of motion with upper and lower joint angle limits of $q_r^u$ and $q_r^l$, that is, the $r$-th element of $\mathbf{q}$ should be within the interval $[q_r^l, q_r^u]$. Let $n$ such joint limit constraints be expressed as $h_2(\mathbf{q}) \leq 0$.

### 6.3.2 Sef-collision and self-occlusion avoidance

We present a self-collision avoidance strategy based on simulating self-collision offline and analyzing the self-collision robot poses. The motivation behind this approach is that joint angles when the robot is in self-collisions depend on the structure of the robot arm as well as the end-

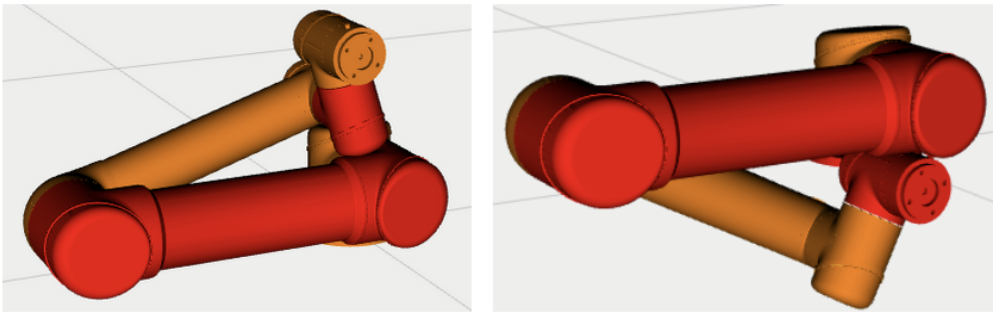


Figure 6.2: Two examples of self-collision of a UR5 robot when the 3rd joint is close to $\pm\pi$.

effector geometries. Joint angles that tend to cause self-collision should be avoided. Thus, we intend to find out the joint angles that frequently cause self-collision and formulate penalties in the optimization objective function to avoid these joint angles. Self-collision can be checked after solving the joint angles. This strategy is inspired by the method proposed in [104]. Two examples of a UR5 in self-collision are shown in Fig. 6.2.

There are two steps in the proposed self-collision avoidance strategy. The first step is to find the joint angles that cause self-collisions. This is done by randomly sampling the configuration space of the robot and checking for self-collision. The second step is to identify the joints and joint angles that cause frequent self-collisions. The joint angles for each joint when the robot is in self-collision are plotted to find potential distribution patterns. If the distribution is even, i.e., no significant peaks, then the joint is less possible to cause self-collisions. Joint angles corresponding to the peaks in the distributions are considered to have higher possibilities to cause self-collision and should be avoided. Assume the robot arm has $n$ joints, $p_i, i = 1, \cdots, n$ peaks are found in the self-collision angles distributions. We use the following loss function to keep robot away from those joint angles:

$$g_{sc} = \sum_{i}^{n} \sum_{p_i}^{p_n} a_1 e^{-b_1(\theta_i - \theta_{p_i})^2} \tag{6.7}$$

where $a_1, b_1$ are the parameters used to control the distribution of the exponential functions, $\theta_{p_i}$ are the joint angles to avoid. Notice that the proposed strategy is to avoid joint angles that can potentially cause self-collisions. Thus, self-collision checking is still needed after solving the

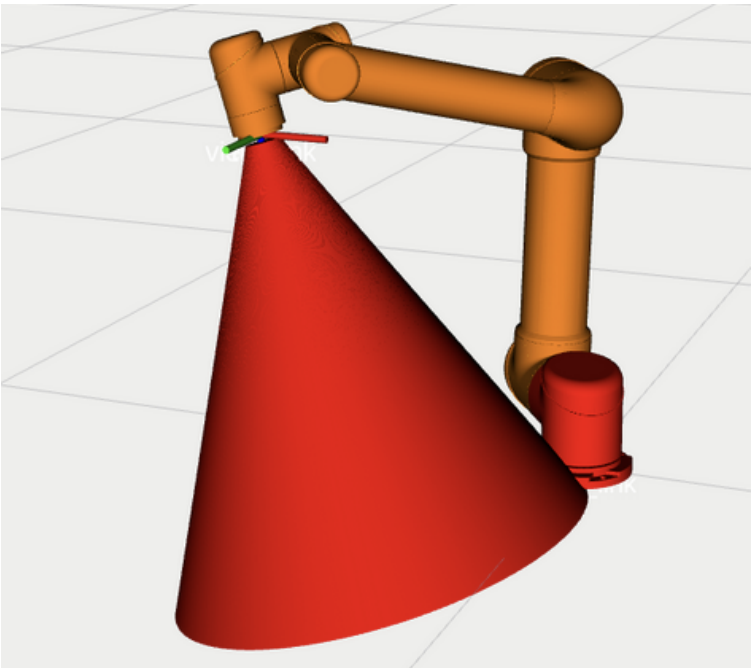


Figure 6.3: An example of self-occlusion of the robotic system. The field-of-view of the sensor is modeled as a cone. The vision sensor is mounted on the robot end-effector. The robot base is in the sensor view at the shown robot pose.

optimization problem.

Self-occlusion can be checked by considering the sensor view as a rigid cone and then checking the robot system for collision. An illustration is provided in Fig. 6.3. The same sampling and analysis strategy for self-collision can be used for self-occlusion. In practice, self-collision and self-occlusion can be combined for a given robotic system. Similar to the self-collision checking analysis, assume the robot arm has $n$ joints, $h_i, i = 1, \cdots, n$ peaks are found in the self-occlusion angles distributions. We use the following loss functions to keep robot away from those joint angles:

$$g_{so} = \sum_{i}^{n} \sum_{h_i}^{h_n} a_2 e^{-b_2(\theta_i - \theta_{h_i})^2} \tag{6.8}$$

where $a_2, b_2$ are the parameters used to control the distribution of the exponential functions, $\theta_{h_i}$ are the joint angles to be avoided.

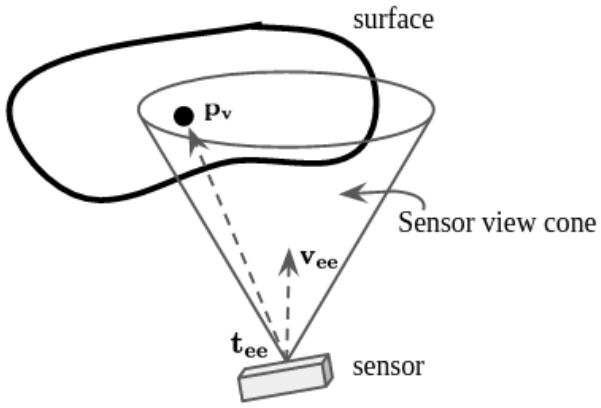



Figure 6.4: The point to be measured, $\mathbf{p}_v$, should remain in the sensor view when the sensor moves to a new view, for which the angle between the vector $\mathbf{p}_v - \mathbf{t}_{ee}$ and $\mathbf{v}_{ee}$ should be constrained.

### 6.3.3 Visibility constraints

Constraints related to object visibility need to be considered, including object-in-view constraint and field-of-distance constraint. In order for a point on the object, $\mathbf{p}_v$, to be in the sensor view, the following condition has to be satisfied (see Fig. 6.4):

$$\cos\left(\frac{\beta_v - \delta_v}{2}\right) \leq \frac{\langle \mathbf{p}_v - \mathbf{t}_{ee}, \mathbf{v}_{ee} \rangle}{|\mathbf{p}_v - \mathbf{t}_{ee}| \cdot |\mathbf{v}_{ee}|} \tag{6.9}$$

To impose this constraint, we choose an angle $\phi$ such that $\phi \leq \frac{1}{2}(\beta_v - \delta_v)$ and consider the constraint $h_{oiv} \leq 0$ where

$$h_{oiv} = \cos\phi - \frac{\langle \mathbf{p}_v - \mathbf{t}_{ee}, \mathbf{v}_{ee} \rangle}{|\mathbf{p}_v - \mathbf{t}_{ee}| \cdot |\mathbf{v}_{ee}|}. \tag{6.10}$$

For quality improvement, one can choose $\mathbf{p}_v$ as the cluster center, i.e., $\mathbf{p}_v = \mathbf{p}_{a_j}$. The constraint that the sensor has to be at a minimum distance away from the object, for both safety and satisfying the field-of-distance requirement can be expressed as:

$$h_{fod} = \max\{d_{fod}, d_s\} - \|\mathbf{t}_{ee} - \mathbf{p}_v\|_2 \leq 0 \tag{6.11}$$

where $d_{fod}$ is the minimum distance for the sensor to measure the object, and $d_s$ is the user specified safety distance between the sensor and the object.

### 6.3.4 The nonlinear optimization formulations

Two objective functions are considered:

$$f_1(\mathbf{q}) = \|\mathbf{q}_0 - \mathbf{q}\|_2 \tag{6.12}$$

and:

$$f_2(\mathbf{q}) = \|\mathbf{q}_0 - \mathbf{q}\|_2 + g_{sc}(\mathbf{q}) + g_{so}(\mathbf{q}) \tag{6.13}$$

The first objective function is to find the robot pose ($\mathbf{q}$) that is closest to the current robot pose for each desired $\mathbf{v} \in V$ such that: (1) the actual sensor view $\mathbf{v}_{ee}$ is aligned with the desired sensor view $\mathbf{v}$ up to an allowed deviation, (2) the object to be measured is in the sensor view when the robot moves to a new pose, and (3) the sensor is at a safe distance from the object. The second objective further penalizes the joint angles that cause self-collision and self-occlusion.

With the above definitions, we formulate the following optimization problem to solve for $\mathbf{q}$:

$$\mathbf{q}^* = \min_{\mathbf{q} \in \mathbb{R}^n} f(\mathbf{q}) \tag{6.14}$$

subject to the constraints:

$$h_1(\mathbf{v}_{ee}, \mathbf{v}) \leq 0 \tag{6.15}$$

$$h_2(\mathbf{q}) \leq 0 \tag{6.16}$$

$$h_{oiv}(\mathbf{v}_{ee}, \mathbf{t}_{ee}, \mathbf{p}_v) \leq 0 \tag{6.17}$$

$$h_{fod}(\mathbf{t}_{ee}, \mathbf{p}_v) \leq 0 \tag{6.18}$$

where $f = f_1$, or $f = f_2$. Solving the above problem gives the robot pose that is closest to the current robot pose for each $\mathbf{v} \in V$. The effectiveness of utilizing $f_1$ and $f_2$ are shown separately in

experiments. The sensor views can be sorted by a predefined priority such that the next-best-view can be chosen as the first solved robot pose corresponding to the prioritized sensor views. For example, the sensor views to improve the point cloud quality can have higher priority over the sensor views to increase the point cloud quantity for symmetric objects or objects that have large surface curvatures. But for objects that have large flat features, sensor views for increasing point cloud quantity in terms of angular spread can be prioritized. Further, multi-threading can be used to speed up the calculation by assigning each sensor view to a thread.

## 6.4 Experiment results

We present results for each of the discussed aspect and component. The system setup is shown in Fig. 5.5. Some of the results are published in [100].

### 6.4.1 Self-collision and self-occlusion analysis

This section presents the self-collision and self-occlusion analysis of a UR5 robot mounted with a Ensenso N35 sensor after applying the proposed method. The sensor view is modelled as a cone with base and height being 0.5 m. 200 thousand robot poses are sampled to find the self-collision and self-occlusion poses, and the corresponding joint angles are recorded.

Figure 6.5 shows the histograms of the angles of the six joints when the robot is in self-collision. The mean and standard deviation of the number of self-collision poses for the six joints are (from the first to the sixth joint): $99.5 \pm 25.6$, $99.5 \pm 22.1$, $99.5 \pm 105.1$, $99.5 \pm 64.5$, $99.5 \pm 30.2$, $99.5 \pm 10.1$. Among the six joints, the first, second, and sixth joint have relatively even distribution, while the third and the fourth joints have two significant peaks. The mean and standard deviation of the two peaks of the third joint are: (-3.12, 0.33), (3.15, 0.33). The mean and standard deviation of the two peaks of the fourth joint are: (-3.14, 1.22), (3.15, 1.24). The sixth joint being the least significant for self-collision is straightforward: no self-collision will arise when rotating the end-effector, thus every joint value for the sixth joint has equal probability of (or nor) causing self-collision. The same explanation applies to the first and second joint. Figure 6.6 further shows the third and the fourth joint angle distributions. The two peaks of the third joint correspond to the

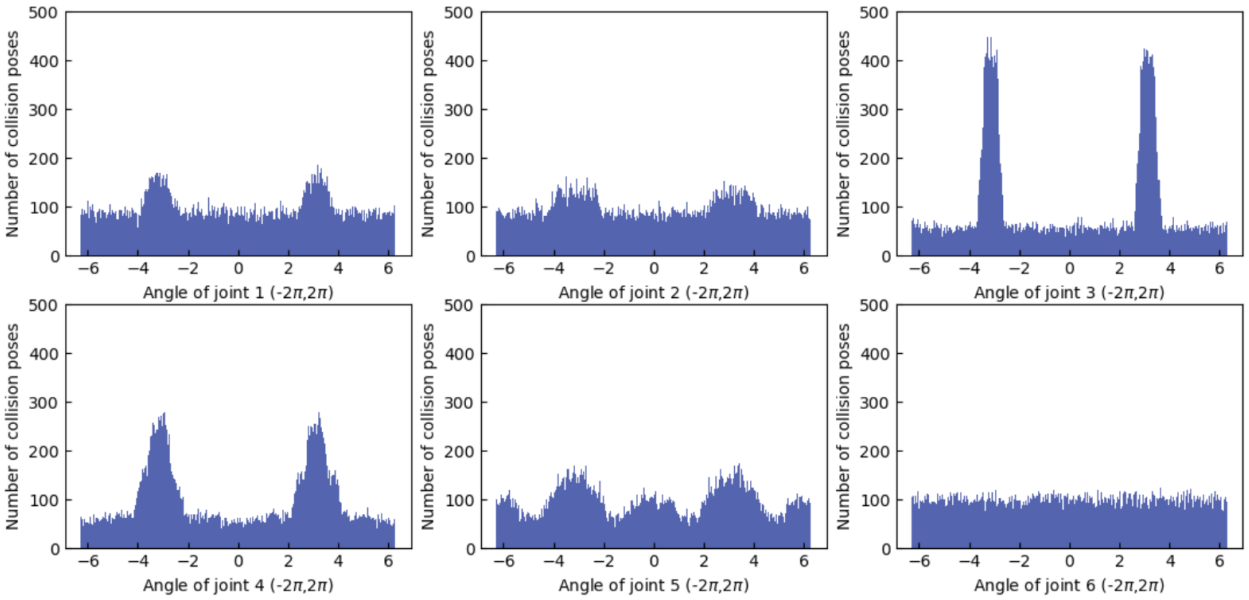



Figure 6.5: Histograms of joint angles of a UR5 robot arm being in self-collision

two configurations shown in Fig. 6.2. Similarly, the two peaks of the fourth joint can be explained in Fig. 6.7, which is due to the mechanical design of the robot arm. Thus, the corresponding joint angles should be avoided.

The joint angle distributions for self-occlusion is shown in Fig. 6.8. The third, fourth, and the fifth joint have significant peaks, and are also fitted using Gaussian distribution. The differences between the less significant joints (such as the first, second and the sixth) and the more significant joints (such as the third, fourth and fifth) can be explained the same way as in self-collision analysis. Notice that the joint angle distributions when the robot is in self-occlusion are subject to change when the sensor is mounted differently.

After identifying the joint angles that should be avoided for self-collision and self-occlusion avoidance, the effectiveness of including penalties in the optimization problem (6.14) is evaluated by choosing the sensor views that will cause self-collision or self-occlusion and checking if the solved robot poses are free of self-collision and self-occlusion. Figure 6.10 shows an example of finding a robot pose that is self-collision free when the sensor view candidate causes self-collision. The penalties of joint angles with high collision probability in objective function (6.13) drive the

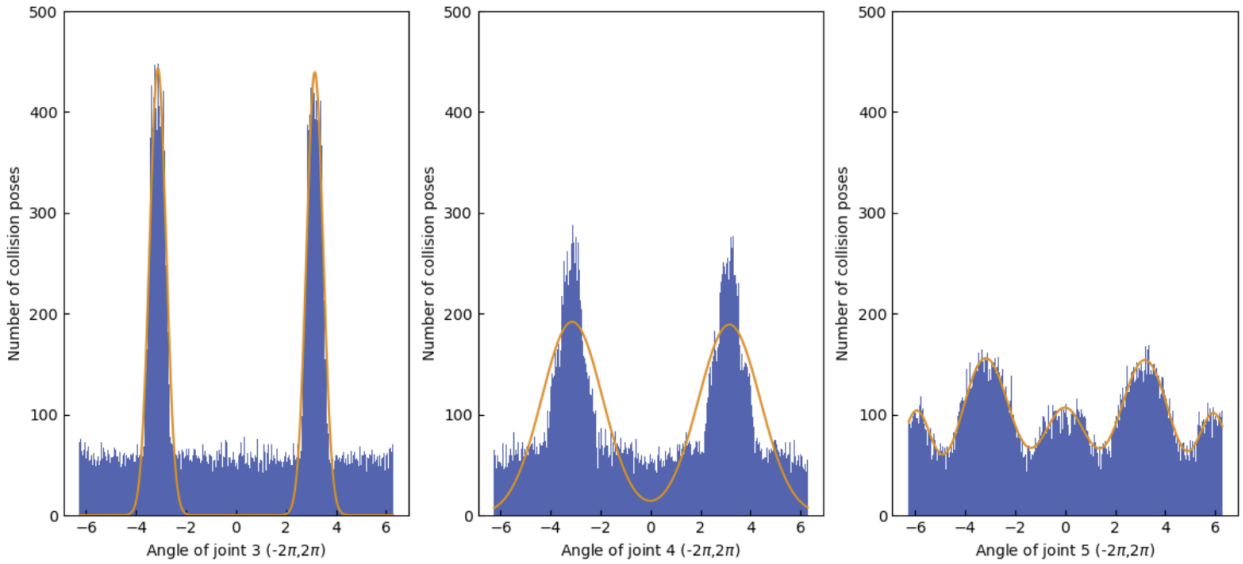



Figure 6.6: Histograms of self-collision joint angles of the third, fourth and fifth joint, which have higher probability of causing self-collisions comparing to other joints. Gaussian distributions are used to find the mean and variance for each peak in the histogram.

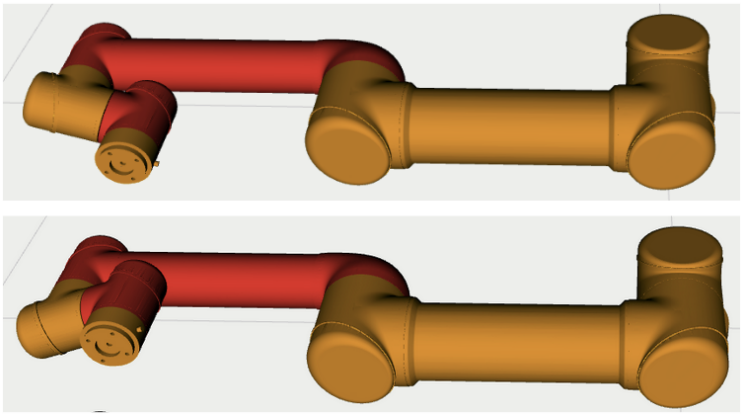


Figure 6.7: Explanation of the two peaks of the fourth joint, shown in Fig. 6.6, which should be avoided.

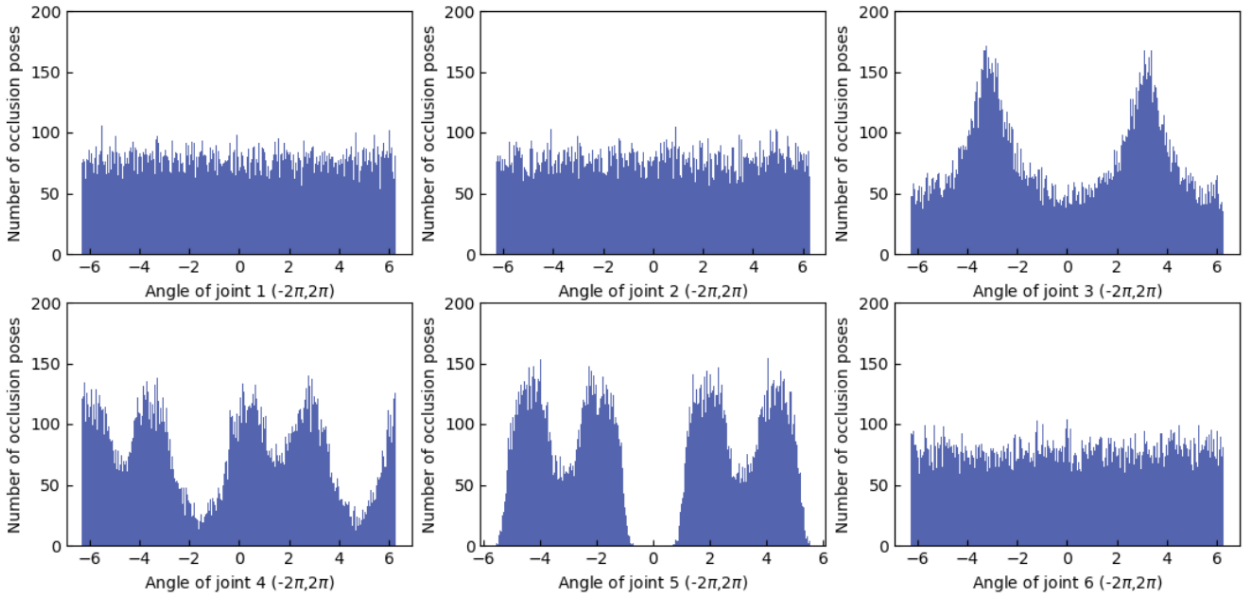



Figure 6.8: Histograms of joint angles that cause self-occlusion during measuring using a vision sensor

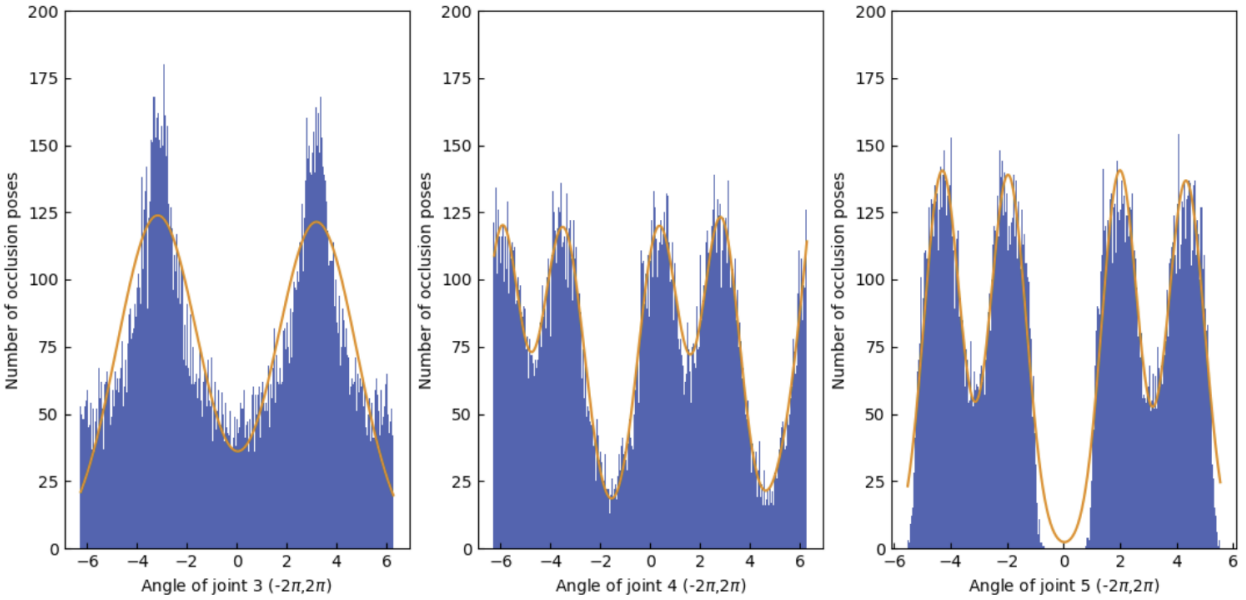



Figure 6.9: Histograms of joint angles that cause self-occlusion. Angles for the third, fourth, and the fifth joints are fitted using Gaussian distributions

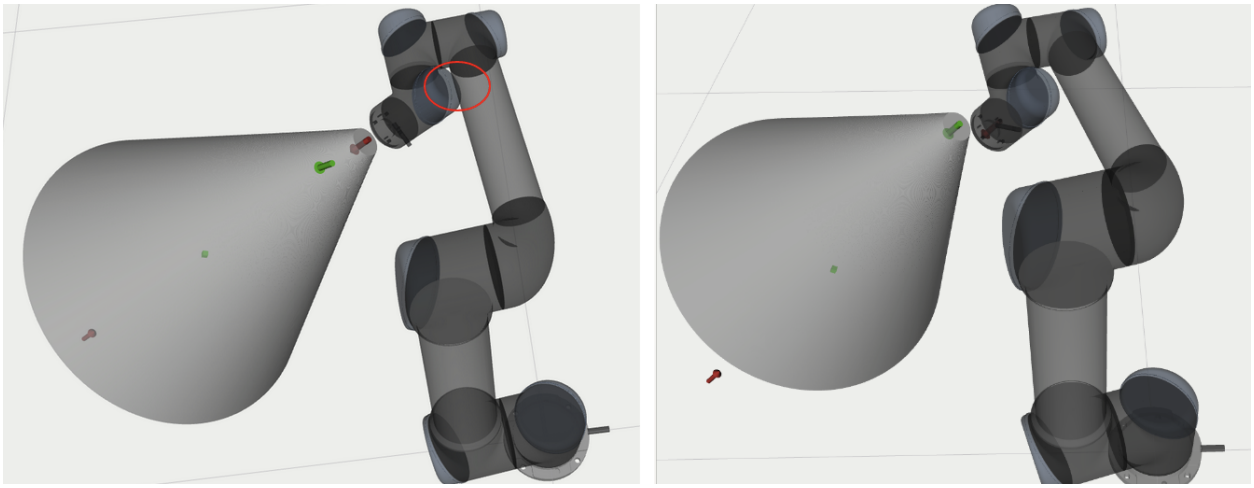


Figure 6.10: Example of avoiding self-collision by penalizing identified joint angles with high collision probability. Left: sensor view candidate (red arrow) that puts the robot in self-collision. Right: the collision-free NBV (green arrow) solved using the proposed optimization formulation.

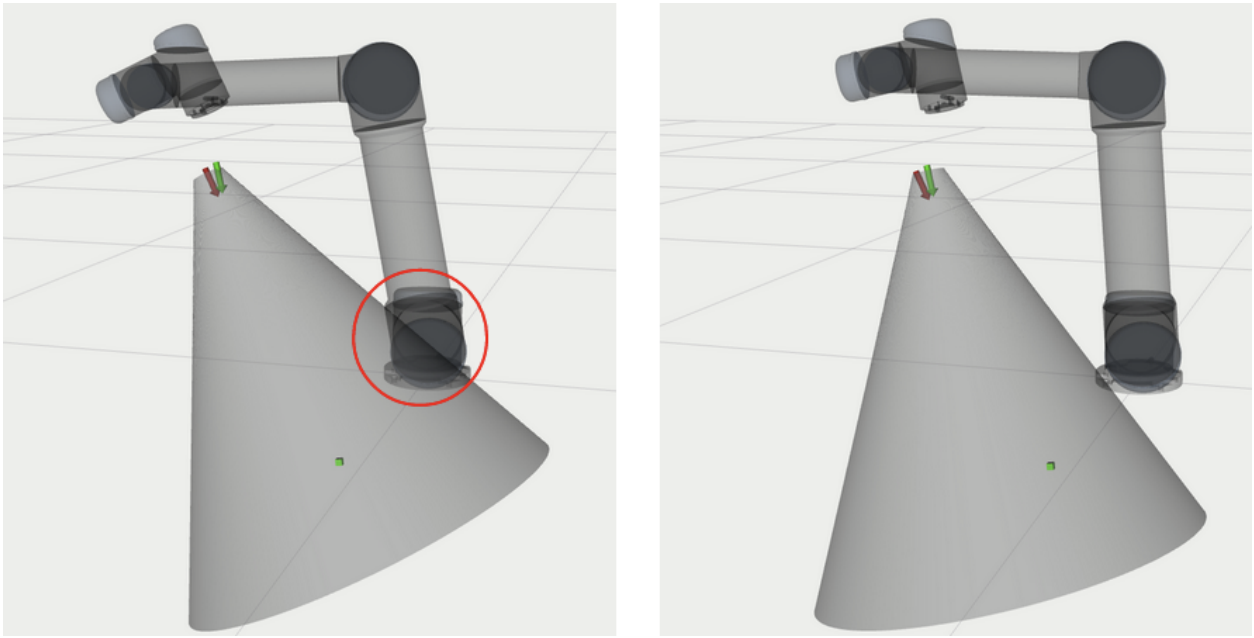


Figure 6.11: Example of avoiding self-occlusion by penalizing identified joint angles with high collision probability. Left: sensor view candidate (red arrow) that puts the robot in self-occlusion. Right: the occlusion-free NBV (green arrow) solved using the proposed optimization formulation.

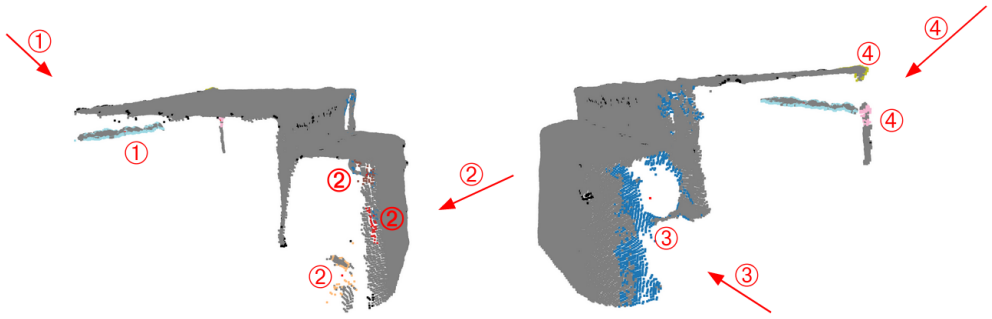



Figure 6.12: The obtained optimal sensor views (red arrows) and the corresponding target areas in the point cloud to be measured by the sensor. The correspondence between the sensor view and the target areas are indicated by the same number. View 1 and 3 each covers one cluster, view 2 covers 3 clusters, and view 4 covers 2 clusters.

robot away from self-collision poses. The effect on self-occlusion avoidance is shown in Fig. 6.11. Notice that when self-occlusion is modeled using rigid body collision, self-occlusion and self-occlusion, and be combined into one checking process.

### 6.4.2 Determination of sensor views

One point cloud of the mechanical component in Fig. 5.6 is selected to evaluate the proposed methods to find a minimum number of sensor views (see Fig. 6.12). The point cloud is representative in the sense that it contains both adjacent outlier clusters and outlier clusters that are far away in terms of the Euclidean distance. When applying the proposed methods, adjacent clusters should be measured with the same sensor view whereas clusters that are far away should be measured from different sensor views. In Fig. 6.12, outliers are grouped into seven clusters. A minimum of four sensor views are obtained to cover all seven clusters; $d_c$ is set to 0.05 m, and $\theta_l^s = 5, \theta_u^s = 65$ degrees. The three clusters that are numbered "2" satisfy both the defined distance and incidence angle constraints and one sensor view is assigned to measure the three clusters simultaneously. The cluster numbered "1" is away from the rest six clusters, so one additional sensor view is generated. The same strategy is applied to clusters and views numbered "3" and "4". Notice that the number of minimum views depend on $d_c$ and $\theta_l^s - \theta_u^s$. For objects with complex geometries and small sizes,

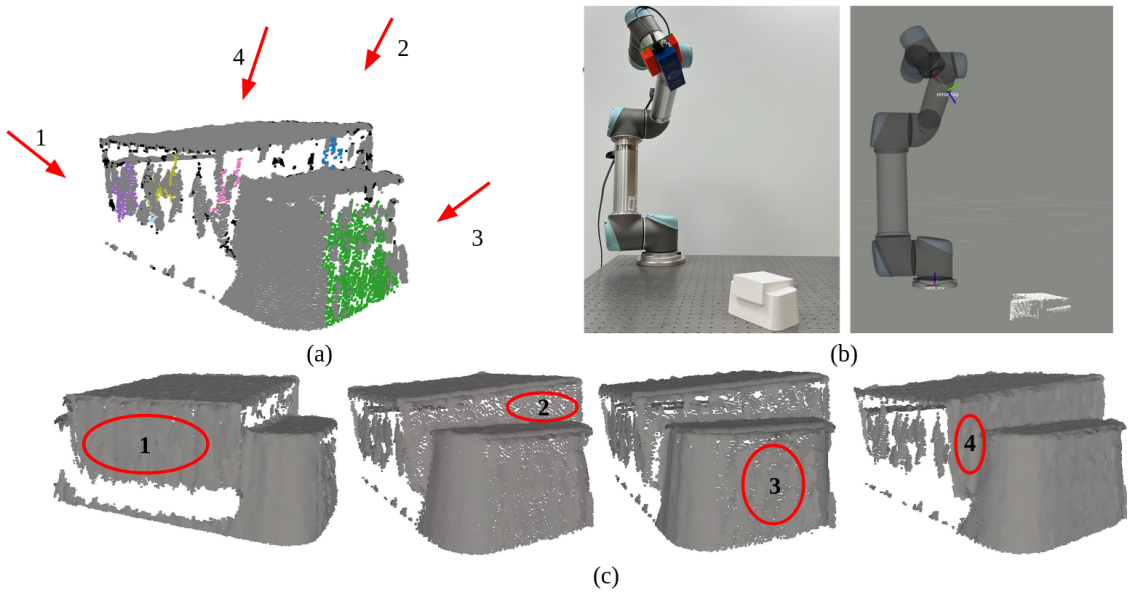



Figure 6.13: Robot poses for quality improvement: (a) four sensor views for quality improvement are shown in red arrows and the identified outliers are colored, (b) the physical robot and its visualization in Rviz for one solved robot pose, (c) the four point clouds collected from the four sensor views.

$d_c$ can be set to a small value to avoid potential occlusions caused by the object. $\theta_l$ and $\theta_u$ can also be set to small values to account for the light conditions and object surface reflections.

### 6.4.3 Robot pose determination corresponding to sensor views

The objective function (6.12) is utilized to calculate the robot poses corresponding to the sensor views. A point cloud collected from a initial sensor view is chosen to evaluate the effectiveness of finding robot poses corresponding to the generated sensor views to improve quality and quantity (see Fig. 6.13 and Fig. 6.14). Five robot poses are solved for five sensor view candidates. Four views to improve the point cloud quality are shown in Figure 6.13, and one view to increase quantity is shown in Figure 6.14. The numbered sensor views and colored clusters in Fig. 6.13(a) match with the numbered and circled clusters in Fig. 6.13(c). The effectiveness of the proposed method can be ascertained from two aspects: (1) for each sensor view, we have solved one robot pose successfully, and (2) the collected point clouds shown in Fig. 6.13(c) indicate significant quality improvement, that is, the holes and irregularities in the four regions are reduced significantly.

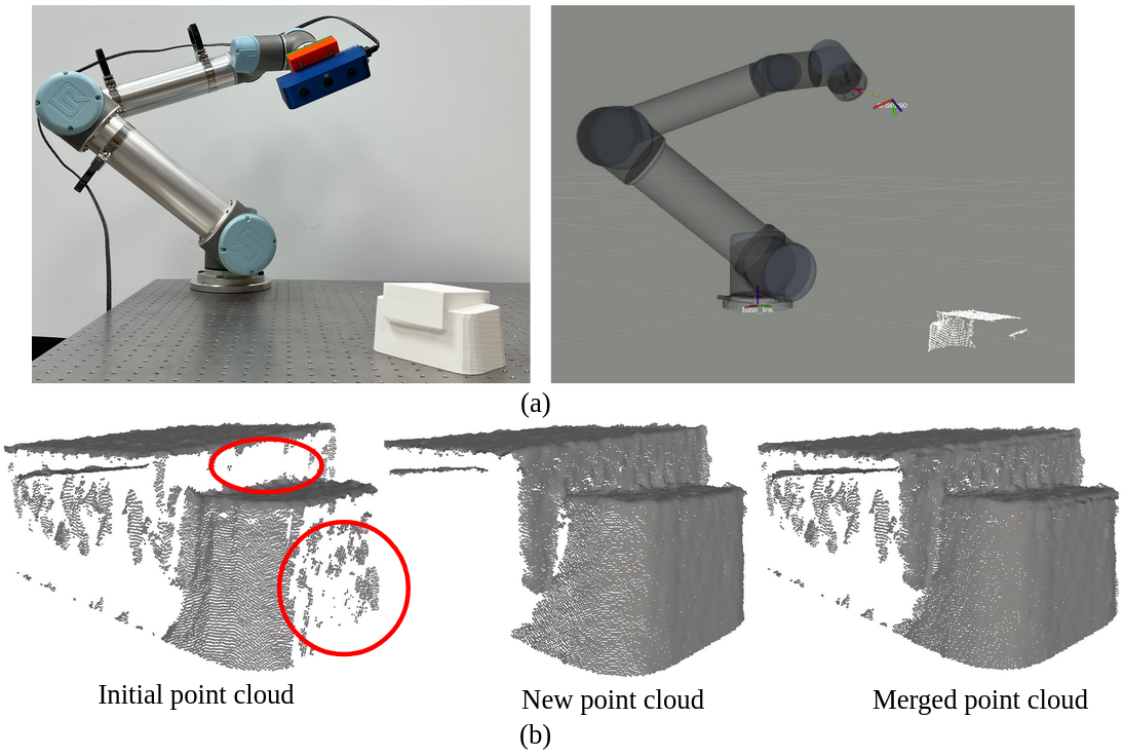



Figure 6.14: Robot pose for quantity improvement: (a) the obtained robot pose, (2) the initial point cloud (left), newly collected point cloud at the solved robot pose (middle), and the merged point cloud (right)

Similarly, a robot pose that corresponds to the sensor view to increase the point cloud quantity is solved successfully, as shown in Figure 6.14. The point cloud collected from the new sensor view indicates the quantity improvement that benefits the pose estimation, as shown in Fig. 6.14(b).

### 6.4.4 Pose estimation comparison

The proposed active perception method is further evaluated by using the changes in pose estimation accuracy. The initial point clouds and the merged point clouds after applying the proposed method are used in the point cloud registration to obtain the object pose. In order to reflect the changes in registration brought by the changes in the point clouds, we choose Go-ICP for registration which gives the global optimal transformations [34]. The fitness and RMSE are used to quantify the registration (pose estimation) accuracy. A sample of the registration results of the mechanical component in Fig. 5.6 is provided in Table 6.1. Six experiments with different initial sensor locations were conducted and point clouds were obtained; increase in fitness is observed for five of the six (sets 2, 3, 4, 5, 6) point clouds after both quantity and quality improvement, and one (Set 1) has decreased fitness after moving to new sensor views. Point cloud sets 2, 3, 4, and 5 have decreased RMSE with data from quantity improvement, and sets 2, 3, and 4 have decreased RMSE with data from quality improvement. The average fitness increase after quantity improvement is 49.3%, and 21.0% increase after quality improvement. The decreased fitness in Set 1 is due to newly introduced outliers. The relative fitness increase depends on various factors: quality and quantity of initial point clouds and geometries and sizes of the objects.

Besides fitness and RMSE, we also observed that point clouds with improved quality require less registration time. One such example is shown in Fig. 6.15(a-c). After moving to the new sensor views for point cloud quality improvement, the registration time is reduced from 86.1 s when using the initial point cloud to an average of 18.5 s when using the updated point clouds from the five sensor views. The changes in registration fitness and RMSE for the gear are very small with the initial fitness being 0.998 and RMSE being 0.85 mm. The small changes in fitness and RMSE can be explained by the geometry of the gear: the concavity and symmetry of the gear teeth make it easier to collect point clouds for pose estimation. The pose estimation results using an impeller are

Table 6.1: Registration fitness and RMSE of the mechanical component

| Data | Fitness | RMSE | Data | Fitness | RMSE |
|---|---|---|---|---|---|
| Set 1 (IN) | 0.999 | 0.65 | Set 4 (IN) | 0.409 | 1.81 |
| Set 1 (QT) | 0.995 | 0.77 | Set 4 (QT) | 0.998 | 0.77 |
| Set 1 (QL) | 0.997 | 0.66 | Set 4 (QL) | 0.428 | 1.76 |
| Set 2 (IN) | 0.920 | 1.10 | Set 5 (IN) | 0.425 | 1.74 |
| Set 2 (QT) | 0.992 | 0.79 | Set 5 (QT) | 0.994 | 0.96 |
| Set 2 (QL) | 0.994 | 0.81 | Set 5 (QL) | 0.996 | 8.8 |
| Set 3 (IN) | 0.914 | 1.25 | Set 6 (IN) | 0.888 | 1.15 |
| Set 3 (QT) | 0.993 | 0.78 | Set 6 (QT) | 0.906 | 1.17 |
| Set 3 (QL) | 0.999 | 0.66 | Set 6 (QL) | 0.909 | 1.15 |
| IN: initial point cloud; QT: point cloud with quantity improvement; QL: point cloud with quality improvement | | | | | |

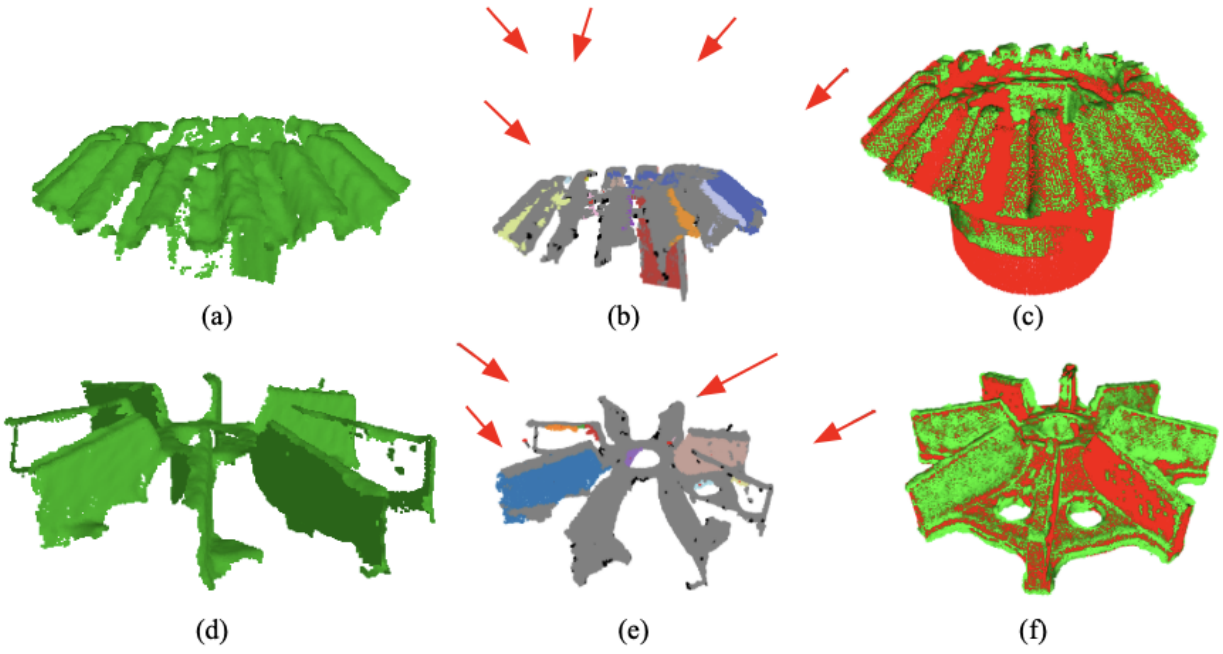



Figure 6.15: Point cloud collection and registration of a gear ((a)-(d)) and impeller ((d)-(e)). (a) and (c) are the initial point clouds, (b) and (c) are the potential sensor views, (c) and (f) are instances of registrations.

shown in Fig. 6.15(d-f). The fitness of registration increased from the initial 0.897 to an average of 0.990 of the four sensor views, and RMSE has decreased from 1.5 mm to an average of 0.87 mm. And the registration time decreased from 22.03 s to 16.88 s.

### 6.4.5 Baseline comparison

The proposed method is compared with two other view planning methods used in reconstruction: random view selection (Baseline 1) and view planning based on unexplored volume (Baseline 2). For Baseline 2, the sensor view is selected based on the information gain calculated using the voxel entropy, which has been proposed and used in [105, 106, 107]. The region around the object is discretized into voxels using Octomap [108]. The occupancy probability of each voxel is updated based on the measured point clouds with the same hit and miss probabilities as defined in [107]. At each step, ray casting is used to estimate the voxels to be seen by each of the 14 views, each view is scored using the utility function based on the entropy of voxel occupancy. The methods from both Baselines choose views from a set of offline selected views. After placing the object in the robot workspace, we manually chose 14 sensor views from which our vision sensor can output point clouds with good quality for a fair comparison. The 14 sensor views also cover all the regions on the object that are visible to the robot. Figure 6.16 shows the selected 14 sensor views selected.

The three methods were compared under two sets of experiments: (1) the robot starts at the same initial location (Location 1), each of the three methods is applied K times to generate sensor views and collect point clouds, the registration fitness of the collected point clouds at each step are compared; (2) the same workflow is applied by using the three methods but the object is rotated by a random angle and translated by a small distance (less than 0.01 m) from the location in (1), denoted as Location 2. The objective of (2) is to test the robustness of the three methods when the object location changes, which happen frequently when working with HMLV parts. Note that in practice, one can stop the data collection when the registration fitness reaches a desired threshold. But to examine the performance of the methods with more data, we generated K views. The selection of K depends on the size of the object. For our experiments, K is set to five.

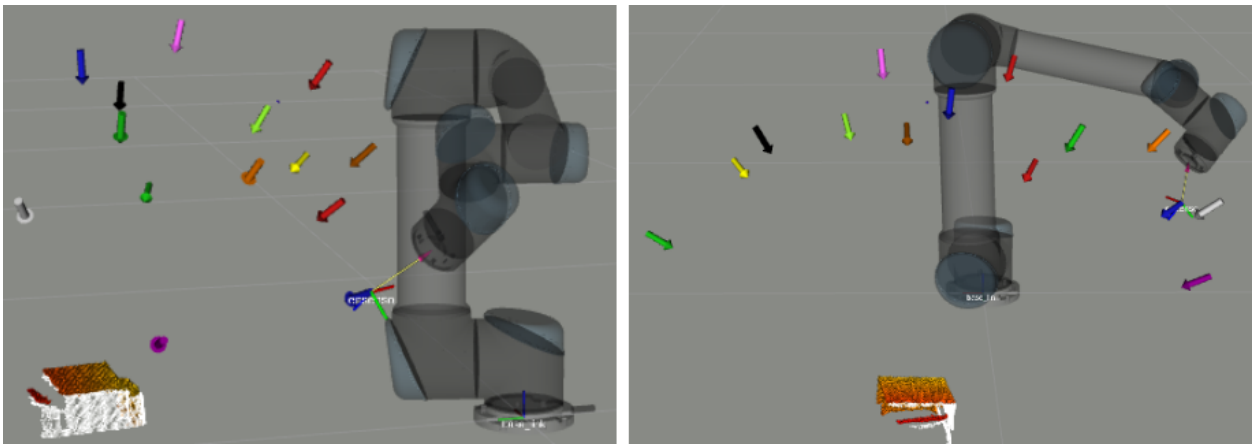


Figure 6.16: The selected 14 sensor views as seen from two different directions.

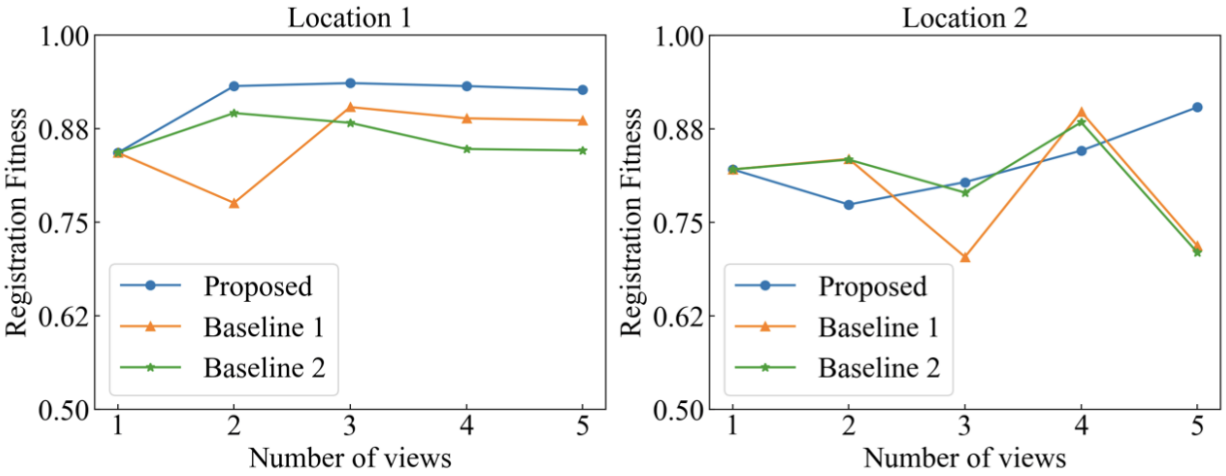



Figure 6.17: Fitness comparison between the proposed method two baselines.

The left figure in Fig. 6.17 shows the registration fitness of the five views when the object is at Location 1. The fitness of the proposed method reaches 0.9 at the second view and maintains at a constant level (around 0.932). The registration fitness at the fifth step of the two Baselines are slightly below that of our method, which indicates observing unexplored volume does not necessarily increase the data that benefits pose estimation. The right figure in Fig. 6.17 shows the registration fitness of the three methods when the object is at Location 2. Our method leads to a fitness of 0.904 at the fifth step while the fitness of the other two methods fluctuated. The fluctuation of the fitness of the Baselines can be explained by the fact that surfaces of the object may not be measured from preferred view angle, which affects the quality of the collected point clouds. But our method can generate views based on the analysis of the current point clouds, thus is more robust, especially when working with HMLV parts. The results of this comparison also highlighted the importance of incorporating a feedback mechanism in generating sensor views and the need to reduce the reliance on offline selected views.

The average computation time for each of the components are the following: (1) point cloud quality analysis: 2.72 s; (2) generate sensor views for point cloud quality improvement: 1.12 s; (3) point cloud quantity analysis and view generation: 0.42 s; (4) solving the inverse kinematics: less than 1 s. Note that the time required for point cloud analysis depends on both the size and the quality of the point cloud. The time required for solving the inverse kinematics depends on the number of sensor views, current robot joint angles, etc.

## 6.5 Conclusions

An active robot perception framework has been proposed for pose estimation in robotic applications. Three key problems associated with this framework have been formulated, and methods to solve them are provided: evaluating the collected point clouds in terms of quality and quantity, determining sensor views to improve the point cloud quality and quantity, and finding robot poses corresponding to the desired sensor views. The effectiveness of the proposed solutions has been shown through various experimental results on three different objects. The point cloud evaluation methods provided potential view directions to collect point clouds. Robot poses that can orient

the sensor along the desired directions to collect better point clouds for pose estimation have been obtained by solving the nonlinear optimization problem. The improvement of the registration accuracy in terms of fitness and RMSE indicated the overall effectiveness of the active perception framework for pose estimation. The comparison of our method with the two Baselines showed that the sensor views generated from our method are more informative for pose estimation, and the method is more robust when working on HMLV tasks.

# 7. CONCLUSIONS AND FUTURE WORK

## 7.1 Conclusions

This dissertation proposed an active robot perception framework to solve the object pose estimation problem. The novelty of the work lies in treating the data collection and pose estimation as an integral part of the developed active perception framework. Point cloud registration is utilized for pose estimation. Data collection for pose estimation has been formulated as a view planning problem. Methods for both pose estimation and data collection have been developed. This dissertation has provided two novel view planning strategies to advance automatic object pose estimation: one requires engineered objects having planes from which data can be obtained and processed and the other based on improving the data for accurate pose estimation without any requirements on the object geometry.

Plane-based point cloud registration and plane-based next-best-view are first developed for engineered objects containing planar geometric features. With the goal of using features that have geometric meanings for registration and designing view planning strategies to collect these geometric features, the proposed registration, and view planning methods have formed a closed loop, i.e., the perception is active. The effectiveness of the proposed registration algorithm has been tested on both convex and concave objects. Simulation and experiment results have shown the effectiveness of the plane-based next-best-views. The same approach can be further extended to develop view planning strategies for other geometric primitives.

To expand the proposed active robot perception framework to objects that do not have planar features, a view planning strategy that is agnostic to object geometry and registration algorithm has been developed. Informative sensor view candidates have been generated to improve the point cloud quality and quantity based on the proposed point cloud analysis methods. Robot poses are determined by solving constrained nonlinear optimization problems instead of checking the voxels in the discretized robot workspace for kinematic feasibility. The effectiveness of the proposed

solutions has been shown through various experimental results on different objects. The improved registration accuracy has validated the effectiveness of the view planning strategy. The advantages of employing the proposed active robot perception for pose estimation are further shown via comparison with two other view planning methods: generated sensor views are more informative for pose estimation, sensor view generation is less dependent on human intervention and more robust to the changes in object location. The proposed active perception framework is flexible in the sense that it can be combined with other point cloud registration methods.

Object pose estimation is fundamental to most robotic applications. Sensing and perception are critical gateways for robots to interact with the real world. Designing active robot perception strategies for object pose estimation has been shown to be both necessary and beneficial.

## 7.2 Future work

There are still many open problems when estimating object pose via active robot perception. Some of the key problems are summarized in the following.

### 7.2.1 Extending plane-based registration and next-best-view planning

The idea behind plane-based registration and next-best-view planning can be potentially extended to include other geometric primitives. Collecting data that can be directly used in the registration algorithm is more efficient for pose estimation.

### 7.2.2 Integrating multiple sensing modalities for pose estimation

Tactile and visual sensing are both important sensing modalities that can be made active. Hybrid motion-force control, which is utilized in the robotic gear chamfering example, can be potentially integrated into the developed active robot perception framework. Contact measurements generate sparse data as opposed to the dense data from visual sensing but is insensitive to light conditions. Combining sensing modalities can potentially be more robust and can estimate object poses with desired levels of accuracy.

### 7.2.3 Learning object geometries

The CAD models of the objects are utilized in pose estimation when registering point clouds. Similar to the idea of collecting point clouds that benefit registration, offline learning using the object CAD models can be potentially used to select regions on the object that lead to more accurate registration results. Thus, informative sensor view candidates for pose estimation can be generated.

### 7.2.4 Planning a sequence of views based on the initial point cloud

The point cloud analysis of quality and quantity is used to generate sensor view candidates. The next-best-view is selected to either improve the quality or the quantity of the collected point clouds. Since multiple sensor view candidates can be generated after analyzing the point cloud, a sequence of sensor views can be planned. The overlap between the sensor views can be predicted by using, for example, ray casting to choose the set of sensor views that have minimal overlap. The obtained views can be further sequenced by formulating the sequencing problem as a Traveling Salesman Problem (TSP) from which one obtains the optimal view sequence.

# REFERENCES


[1] K. C. Sahoo and C. H. Menq, "Localization of 3-D objects having complex sculptured surfaces using tactile sensing and surface description," *Journal of Manufacturing Science and Engineering, Transactions of the ASME*, vol. 113, no. 1, pp. 85–92, 1991.

[2] S. Y. Chen and Y. F. Li, "Automatic Sensor Placement for Model-Based Robot Vision," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, pp. 393–408, 2 2004.

[3] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[4] B. Siciliano and O. Khatib, *Springer handbook of robotics*. springer, 2016.

[5] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[6] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, 2021.

[7] A. Gameros, S. Lowth, D. Axinte, A. Nagy-Sochacki, O. Craig, and H. R. Siller, "State-of-the-art in fixture systems for the manufacture and assembly of rigid components: A review," 12 2017.

[8] L. Shao and R. A. Volz, "Methods and strategies of object localization," in *Proceedings of the NASA Conference on Space Telerobotics*, p. 229, National Aeronautics and Space Administration, 1989.

[9] Z. Xiong, M. Y. Wang, and Z. Li, "A Near-Optimal Probing Strategy for Workpiece Localization," *IEEE Transactions on Robotics*, vol. 20, no. 4, pp. 668–676, 2004.

[10] G. Schleth, A. Kuss, and W. Kraus, "Workpiece localization methods for robotic welding – a review," *50th International Symposium on Robotics, ISR 2018*, pp. 50–55, 2018.

[11] O. D. Faugeras and M. Hebert, "A 3-D recognition and positioning algorithm using geometrical matching between primitive surfaces,"

[12] O. D. Faugeras and M. Hebert, "The Representation, Recognition, and Locating of 3-D Objects," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 27–52, 1986.

[13] Z. Li, J. Gou, and Y. Chu, "Geometric algorithms for workpiece localization," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 864–878, 1998.

[14] K. T. Gunnarsson and F. B. Prinz, "CAD Model-Based Localization of Parts in Manufacturing," *Computer*, vol. 20, no. 8, pp. 66–74, 1987.

[15] W. E. L. Grimson and T. Lozano-Pérez, "Model-Based Recognition and Localization from Sparse Range or Tactile Data," *The International Journal of Robotics Research*, vol. 3, no. 3, pp. 3–35, 1984.

[16] W. E. L. Grimson and T. Lozano-Pérez, "Localizing Overlapping Parts by Searching the Interpretation Tree," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 4, pp. 469–482, 1987.

[17] S. J. Gordon and W. P. Seering, "Realtime Part Position Sensing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 374–386, 1988.

[18] R. C. Bolles and P. Horaud, "3DPO: A Three-Dimensional Part Orientation System," tech. rep.

[19] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, pp. 1–10, 3 1966.

[20] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[21] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, p. 629, 4 1987.

[22] N. J. Higham, "The symmetric procrustes problem," Tech. Rep. I988.

[23] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America A*, vol. 5, p. 1127, 7 1988.

[24] S. Umeyama, "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 376–380, 1991.

[25] M. W. Walker, L. Shao, and R. A. Volz, "Estimating 3-D location parameters using dual number quaternions," *CVGIP: Image Understanding*, vol. 54, pp. 358–367, 11 1991.

[26] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-D rigid body transformations: A comparison of four major algorithms," *Machine Vision and Applications*, vol. 9, no. 5-6, pp. 272–290, 1997.

[27] Z. Xiong, M. Y. Wang, and Z. Li, "A computer-aided probing strategy for workpiece localization," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3941–3946, 2003.

[28] L. M. Zhu, H. G. Luo, and H. Ding, "Optimal measurement point planning for workpiece localization," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, pp. 1562–1567, 2004.

[29] B. Saund, S. Chen, and R. Simmons, "Touch based localization of parts for high precision manufacturing," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 378–385, Institute of Electrical and Electronics Engineers Inc., 7 2017.

[30] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[31] S. Bouaziz, A. Tagliasacchi, and M. Pauly, "Sparse iterative closest point," in *Computer graphics forum*, vol. 32, pp. 113–123, Wiley Online Library, 2013.

[32] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp.," in *Robotics: science and systems*, vol. 2, p. 435, Seattle, WA, 2009.

[33] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings third international conference on 3-D digital imaging and modeling*, pp. 145–152, IEEE, 2001.

[34] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-icp: A globally optimal solution to 3d icp point-set registration," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 11, pp. 2241–2254, 2015.

[35] G. Izatt, H. Dai, and R. Tedrake, "Globally optimal object pose estimation in point clouds with mixed-integer programming," in *Robotics Research*, pp. 695–710, Springer, 2020.

[36] J. Hu, P. R. Pagilla, and S. Darbha, "A novel method for the localization of convex work-pieces in robot workspace using gauss map," in *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 2143–2148, 2021.

[37] H. Maron, N. Dym, I. Kezurer, S. Kovalsky, and Y. Lipman, "Point registration via efficient convex relaxation," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–12, 2016.

[38] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[39] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE international conference on robotics and automation*, pp. 3212–3217, IEEE, 2009.

[40] S. Salti, F. Tombari, and L. Di Stefano, "Shot: Unique signatures of histograms for surface and texture description," *Computer Vision and Image Understanding*, vol. 125, pp. 251–264, 2014.

[41] E. Wahl, U. Hillenbrand, and G. Hirzinger, "Surflet-pair-relation histograms: a statistical 3d-shape representation for rapid classification," in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pp. 474–481, IEEE, 2003.

[42] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 998–1005, Ieee, 2010.

[43] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1802–1811, 2017.

[44] Z. J. Yew and G. H. Lee, "3dfeat-net: Weakly supervised local 3d features for point cloud registration," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 607–623, 2018.

[45] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3d point cloud matching with smoothed densities," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5545–5554, 2019.

[46] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8958–8966, 2019.

[47] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3523–3532, 2019.

[48] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust & efficient point cloud registration using pointnet," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7163–7172, 2019.

[49] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[50] Z. Y. Jia, J. W. Ma, F. J. Wang, and Y. M. Ding, "Investigation of a measurement scheme based on IGES," *Measurement: Journal of the International Measurement Confederation*, vol. 47, pp. 658–668, 1 2014.

[51] R. Bajcsy, "Active Perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.

[52] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, 2018.

[53] W. R. Scott, G. Roth, and J.-F. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Computing Surveys (CSUR)*, vol. 35, no. 1, pp. 64–96, 2003.

[54] P. S. Blaer and P. K. Allen, "Data acquisition and view planning for 3-d modeling tasks," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 417–422, IEEE, 2007.

[55] J. I. Vasquez-Gomez, L. E. Sucar, and R. Murrieta-Cid, "View planning for 3d object reconstruction with a mobile manipulator robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4227–4233, IEEE, 2014.

[56] R. Monica, J. Aleotti, and S. Caselli, "A kinfu based approach for robot spatial attention and view planning," *Robotics and Autonomous Systems*, vol. 75, pp. 627–640, 2016.

[57] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3d reconstruction," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3477–3484, IEEE, 2016.

[58] Z. Meng, H. Qin, Z. Chen, X. Chen, H. Sun, F. Lin, and M. H. Ang, "A two-stage optimized next-view planning framework for 3-d unknown environment exploration, and structural reconstruction," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1680–1687, 2017.

[59] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon path planning for 3d exploration and surface inspection," *Autonomous Robots*, vol. 42, no. 2, pp. 291–306, 2018.

[60] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient autonomous exploration planning of large-scale 3-d environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1699–1706, 2019.

[61] E. Vidal, N. Palomeras, K. Istenič, N. Gracias, and M. Carreras, "Multisensor online 3d view planning for autonomous underwater exploration," *Journal of Field Robotics*, vol. 37, no. 6, pp. 1123–1147, 2020.

[62] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pp. 146–151, IEEE, 1997.

[63] F. A. R. Martins, J. G. García-Bermejo, E. Zalama, and J. R. Perán, "An optimized strategy for automatic optical scanning of objects in reverse engineering," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 217, pp. 1167–1171, 8 2003.

[64] F. A. R. Martins, J. G. García-Bermejo, E. Z. Casanova, and J. R. Perán González, "Automated 3D surface scanning based on CAD model," *Mechatronics*, vol. 15, pp. 837–857, 9 2005.

[65] N. A. Massios, R. B. Fisher, *et al.*, *A best next view selection algorithm incorporating a quality criterion*, vol. 2. Department of Artificial Intelligence, University of Edinburgh, 1998.

[66] C. Mehdi-Souzani, F. Thiébaut, and C. Lartigue, "Scan planning strategy for a general digitized surface," 2006.

[67] A. E. Johnson, R. Hoffman, J. Osborn, and M. Hebert, "A system for semi-automatic modeling of complex environments," in *Proceedings. International Conference on Recent Advances in 3-D Digital Imaging and Modeling (Cat. No. 97TB100134)*, pp. 213–220, IEEE, 1997.

[68] E. Zussman, H. Schuler, and G. Seliger, "Advanced manufacturing Technolo lu Analysis of the Geometrical Features Detectability Constraints for Laser-Scanner Sensor Planning," tech. rep.

[69] K. Tarabanis, R. Y. Tsai, and P. K. Allen, "Analytical Characterization of the Feature Detectability Constraints of Resolution, Focus, and Field-of-View for Vision Sensor Planning," *CVGIP: Image Understanding*, vol. 59, pp. 340–358, 5 1994.

[70] T. C. Woo and B. F. von Turkovich, "Visibility Map and Its Application to Numerical Control," *CIRP Annals - Manufacturing Technology*, vol. 39, pp. 451–454, 1 1990.

[71] R. M. Bolle and D. B. Cooper, "On Parallel Bayesian Estimation And Recognition For Large Data Sets, With Application To Estimating 3-D Complex-Object Position From Range Data," in *Computer Vision for Robots* (O. D. Faugeras and R. B. Kelley, eds.), vol. 0595, p. 90, SPIE, 6 1986.

[72] R. Eidenberger, T. Grundmann, W. Feiten, and R. Zoellner, "Fast parametric viewpoint estimation for active object detection," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 309–314, 2008.

[73] W. Sheng, N. Xi, M. Song, and Y. Chen, "Graph-based surface merging in CAD-guided dimensional inspection of automotive parts," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3127–3132, 2001.

[74] D. Bazazian, J. R. Casas, and J. Ruiz-Hidalgo, "Fast and robust edge extraction in unorganized point clouds," in *2015 international conference on digital image computing: techniques and applications (DICTA)*, pp. 1–8, IEEE, 2015.

[75] C. Weber, S. Hahmann, and H. Hagen, "Sharp feature detection in point clouds," in *2010 Shape Modeling International Conference*, pp. 175–186, IEEE, 2010.

[76] A. Boulch and R. Marlet, "Fast and robust normal estimation for point clouds with sharp features," in *Computer graphics forum*, vol. 31, pp. 1765–1774, Wiley Online Library, 2012.

[77] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proceedings of the 19th annual conference on computer graphics and interactive techniques*, pp. 71–78, 1992.

[78] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[79] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Sra, and G. Ridgeway, "Clustering on the unit hypersphere using von mises-fisher distributions.," *Journal of Machine Learning Research*, vol. 6, no. 9, 2005.

[80] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*, pp. 321–352, Springer, 2005.

[81] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[82] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," 1955.

[83] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 928–935, IEEE, 2015.

[84] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[85] B. H. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial optimization*, vol. 1. Springer, 2011.

[86] M. H. Raibert and J. J. Craig, "Hybrid position/force control of manipulators," 1981.

[87] J. Hu and P. R. Pagilla, "Dual-edge robotic gear chamfering with registration error compensation," *Robotics and Computer-Integrated Manufacturing*, vol. 69, p. 102082, 2021.

[88] J. Hu and P. R. Pagilla, "A novel force and motion control strategy for robotic chamfering of gears," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8710–8715, 2020.

[89] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[90] B. K. P. Horn, "Extended gaussian images," *Proceedings of the IEEE*, vol. 72, no. 12, pp. 1671–1686, 1984.

[91] A. Zonca, L. Singer, D. Lenz, M. Reinecke, C. Rosset, E. Hivon, and K. Gorski, "healpy: equal area pixelization and spherical harmonics transforms for data on the sphere in python," *Journal of Open Source Software*, vol. 4, p. 1298, Mar. 2019.

[92] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems," *Mathematical programming*, vol. 10, no. 1, pp. 147–175, 1976.

[93] J. Saunderson, P. A. Parrilo, and A. S. Willsky, "Semidefinite descriptions of the convex hull of rotation matrices," *SIAM Journal on Optimization*, vol. 25, p. 1314–1343, Jan 2015.

[94] J. Yang, Z. Cao, and Q. Zhang, "A fast and robust local descriptor for 3d point cloud registration," *Information Sciences*, vol. 346, pp. 163–179, 2016.

[95] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European conference on computer vision*, pp. 766–782, Springer, 2016.

[96] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, pp. 189–206, 4 2013.

[97] J. Hu, P. R. Pagilla, and S. Darbha, "The next-best-view for workpiece localization in robot workspace," in *2021 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1201–1206, 2021.

[98] A. Makhal and A. K. Goins, "Reuleaux: Robot base placement by reachability analysis," in *2018 Second IEEE International Conference on Robotic Computing (IRC)*, pp. 137–142, IEEE, 2018.

[99] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *kdd*, vol. 96, pp. 226–231, 1996.

[100] J. Hu and P. R. Pagilla, "View planning for object pose estimation using point clouds: an active robot perception approach," *IEEE Robotics and Automation Letters*, 2022. accepted.

[101] H. Nagarajan, M. Lu, S. Wang, R. Bent, and K. Sundar, "An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs," *Journal of Global Optimization*, 2019.

[102] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff, *Pyomo–optimization modeling in python*, vol. 67. Springer Science & Business Media, third ed., 2021.

[103] W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in python," *Mathematical Programming Computation*, vol. 3, no. 3, pp. 219–260, 2011.

[104] D. Rakita, B. Mutlu, and M. Gleicher, "Relaxedik: Real-time synthesis of accurate and feasible robot arm motion.," in *Robotics: Science and Systems*, pp. 26–30, Pittsburgh, PA, 2018.

[105] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, "Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects," *Journal of Real-Time Image Processing*, vol. 10, no. 4, pp. 611–631, 2015.

[106] J. Delmerico, S. Isler, R. Sabzevari, and D. Scaramuzza, "A comparison of volumetric information gain metrics for active 3d object reconstruction," *Autonomous Robots*, vol. 42, no. 2, pp. 197–208, 2018.

[107] M. Lauri, J. Pajarinen, J. Peters, and S. Frintrop, "Multi-sensor next-best-view planning as matroid-constrained submodular maximization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5323–5330, 2020.

[108] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013. Software available at `https://octomap.github.io`.