Rectangular-shaped object recognition and pose estimation

Thanh Long Vu, Liyang Liu, Gavin Paul, Teresa Vidal-Calleja
Centre for Autonomous Systems
University of Technology Sydney, Australia
15 Broadway, Ultimo, NSW 2007, Australia
Thanh.L.Vu@student.uts.edu.au, {Liyang.Liu, Gavin.Paul, Teresa.Vidalcalleja}@uts.edu.au

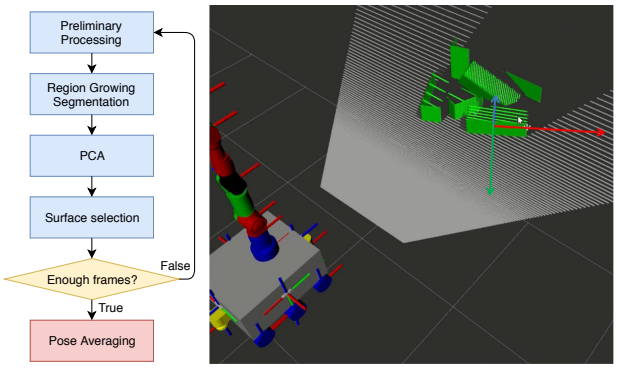
Abstract

This paper presents a novel solution for rectangular-shaped object pose estimation in the robotic bin-picking problem, using data from a single RGB-D camera collecting point cloud data from a fixed position. The key benefit of the presented framework is its ability to accurately and robustly locate an object position and orientation, which allows for high-precision robotic grasping and placing of such objects in an open-loop motion execution system. Firstly, intelligent grasping surface selection is performed, then Principal Component Analysis is used for pose estimation and finally, rotation averaging is integrated to significantly improve noise-reduction over time. Comparisons between the resulting poses and ones estimated by a traditional Iterative Closest Point technique, have demonstrated the framework's advantages for pose estimation tasks.

1 Introduction

Recent years have seen increased interest in autonomous field operation, aided by high-precision perception capabilities [Paul *et al.*, 2016b] [Paul *et al.*, 2016a] [Paul *et al.*, 2013]. The work presented in this paper has primarily been motivated by the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) 2020, to be held in Abu Dhabi, UAE in February 2020. A fundamental aim in MBZIRC is for an Unmanned Ground Vehicle (UGV) to autonomously build a pre-designed structure in a short time frame, which requires the demonstration of robust and precise robotic pick-and-place capabilities.

This paper illustrates a perception solution for the UGV to accurately estimate object shape for grasping and releasing from a single viewing perspective for an open-loop motion execution system. The solution proposed involves identifying the surface of a single rectangular prism-shaped brick within a pile of unorganised bricks of predefined colour and size, and performing highly accurate pose estimation and noise reduction by exploiting a novel series of algorithms.



(a) (b)

Figure 1: Proposed framework pipeline and its demonstration. (a) Pose estimation pipeline. (b) Simulation of an arm-equipped robot detecting the best brick for picking from a pile, chosen brick is labeled with RGB-axes.

In the surface detection and segmentation stage, Principal Component Analysis (PCA) is applied to the point cloud data collected by the RGB-D camera. PCA is a well-known technique used in dimensionality reduction, feature extraction and distribution visualisation [Bishop, 2006]. The technique has been widely applied in various scientific fields to accomplish numerous tasks such as two-dimensional face recognition [Yang *et al.*, 2004], genome-wide single-nucleotide polymorphism data analysis [Abraham and Inouye, 2014], and robotic hands posture and coordination synergies [Brown and Asada, 2007].

Due to the wall-building objectives of the MBZIRC robot, the estimated pose of the brick must match the center and natural orientation of the brick surface for accurate picking. Knowledge of this pose can help with the planning of a firm and robust grasping approach. Consequently, accurate object placement for building construction becomes more straightforward. The added benefit of accurate pose estimation is that it allows for a less-

constrained mechanical gripper design. On the one hand, due to development time constraint, the robot grasping mechanism offers limited error tolerance for successful attachment. High-precision pose estimate becomes a relatively cheaper option in this design activity. On the other hand, data obtained from depth sensors typically contains large amounts of noise, which may become the major hindrance in achieving the system goal. For outdoor applications, especially in middle-eastern countries where sunlight is particularly strong, sensor noise may appear especially pronounced. The reason is that sunlight consists of a strong infra-red (IR) light component that can severely interfere with a camera's IR stereo system. PCA generated brick surface orientation estimates will thus exhibit non-negligible variation from frame to frame. To limit fluctuating pose problems due to noise, multiple depth data sets can be collected over a short time duration in which pose estimation is performed for each time frame. By then applying a novel averaging method it is possible to achieve the best estimate of face orientation. Rotation averaging is thus a key component of the presented high-precision pose estimation framework.

1.1 Related Work

Previous works exploiting PCA-based feature extraction include, pulsed eddy current responses [Sophian *et al.*, 2003], change detection of multidimensional data [Kuncheva and Faithfull, 2013], and colour retinal images [Li and Chutatape, 2004]. The proposed solution involves executing PCA feature extraction for three-dimensional point cloud data.

To address the challenge of orientation robustness, earlier works have proposed rotation averaging least square solutions [Gamage and Lasenby, 2002], rotation averaging for DNA helix [Wang, 1969], and large-scale rotation averaging [Chatterjee and Madhav Govindu, 2013]. [Hartley *et al.*, 2013] provided a thorough review on rotation averaging methods.

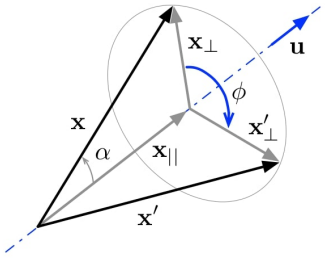
The methods can be categorised by the representation domain they operate on. Rotations can be represented in the angle-axis form in the Euclidean domain,

$$\mathbf{r} = \phi\mathbf{u}, \qquad \phi \in \mathbb{R},\ \mathbf{u} \in \mathbb{R}^3,\ \mathbf{u}\cdot\mathbf{u}^T = 1 \tag{1}$$

where vector $\mathbf{r}$ is a rotation of angle ϕ about an axis in $\mathbf{u}$. The definition is illustrated in Figure 2 [Sol, 2017]. Averaging boils down to finding the algebraic mean for $\mathbf{v}$. Rotations can also be represented in the Manifold domain of $\mathbb{SO}(3)$ (the group of all 3-dimensional rotations), expressed as 3×3 rotation matrix R or unit quaternion, $\mathbf{q}$. The unit quaternion is defined as,

$$\mathbf{q} = (\cos(\theta), \mathbf{u}\sin(\theta)), \qquad \theta = \frac{1}{2}\phi \tag{2}$$

The set of all unit quaternions form a unit sphere, S^3 in $\mathbb{R}^4$. Figure 3 [Sol, 2017] illustrates rotation in the unit-quaternion form. Averaging can be done as a Chordal L_2-mean in the $\mathbb{SO}(3)$ domain.



$$\mathbf{x} = \mathbf{x}_{||} + \mathbf{x}_\perp$$
$$\mathbf{x}_{||} = \mathbf{u}\,\mathbf{u}^\top\mathbf{x}$$
$$\mathbf{x}_\perp = \mathbf{x} - \mathbf{u}\,\mathbf{u}^\top\mathbf{x}$$

Figure 2: Rotation of a vector, $\mathbf{x}$ about the axis, $\mathbf{u}$ by an angle, ϕ [Sol, 2017].

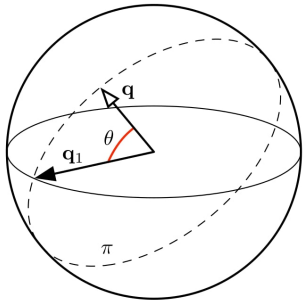


Figure 3: In unit 3-sphere manifold, quaternion $\mathbf{q}$ defines an angle, $\theta = \frac{1}{2}\phi$ with a unit quaternion, $\mathbf{q}_1$ [Sol, 2017].

The angle-axis rotation representation is inherently error-prone as it allows representation ambiguity. Multiple angle-axis values can all represent the same orientation. For example, the aforementioned angle-axis, $\mathbf{r}$ can also be written as $(2\pi - \phi)(-\mathbf{u})$. As another example, a random angle, ϕ with a zero mean may show a small magnitude due to noise: $0 < \phi < \epsilon$. Similarly, with a small perturbation, it can swing in the opposite direction and approach the maximum angle, $\phi \to 2\pi$. Averaging over such values will generally lead to a meaningless orientation. For this reason, the $\mathbb{SO}(3)$ form, due to its unique representation, has been chosen in the proposed framework. In implementation, we use the unit quaternion for rotation parameterisation in $\mathbb{SO}(3)$ and perform averaging by minimising the chordal error.

Alternatively, rotation averaging can also be viewed from the perspective of the optimisation method in use. Some $\mathbb{SO}(3)$ based methods may take an iterative form to search where many steps are required to achieve solution convergence. There are also more elegant methods that exploit simple linear algebra techniques if a closed-form solution is theoretically guaranteed. For simplicity and accuracy, this framework incorporates the simple "Chordal L_2-Mean method"[Hartley *et al.*, 2013].

1.2 Overview

The proposed framework can be described as a pipeline that processes point cloud data received from an RGB-D camera to robustly estimate the pose of a rectangular-shaped object. The collected point cloud is first simplified with a voxel grid filter [He *et al.*, 1995]. Then, it is filtered by distance, colour and number of points. The filtered cloud is segmented into clusters of separate surfaces, and PCA is applied onto these clusters for the purposes of object recognition and pose estimation. Finally, and most importantly, rotation averaging is applied to the PCA outputs collected over multiple frames for result stabilisation and noise reduction.

The remainder of this paper is structured as follows. Section 2 describes region growing segmentation and PCA, followed by the details of the proposed framework. Section 3 documents the comparison between the proposed solution and the Iterative Closest Points (ICP) pose estimation method. The outcomes of Section 3 are discussed in Section 4. Section 5 then concludes with a summary of the proposed pipeline as well as potential future research and development.

2 Methodology

This section first introduces the region growing segmentation for point clouds and details the Principal Component Analysis (PCA) that is used to determine the primary axes. This is followed by a description of the proposed framework for rectangular surface recognition and pose estimation. The final subsection contains an explanation of the rotation averaging method and details of how it is integrated into the proposed framework.

2.1 Region Growing Segmentation

Region growing is a class of algorithm which is best known for its "split and merge technique" [Adams and Bischof, 1994]. Region growing might be used on point cloud data to extract clusters of surface regions [Vo *et al.*, 2015].

Given a point cloud, P, the point, $\mathbf{p}_i$ has a normal vector, $\vec{\boldsymbol{n}}_i$ and a curvature value, c_i. All normal vectors are stored in the point normal set, $\mathbb{N}$ and curvature values in the point curvature set, $\mathbb{C}$.

Let $\mathbb{R}$ be the set of all regions extracted from the point cloud and $\mathbb{A}$ be the remaining points. Initially, $\mathbb{R} = \varnothing$ and $\mathbb{A} = \mathbb{P} \setminus \mathbb{R}$.

As long as $\mathbb{A} \neq \varnothing$, region growing will choose the starting point, p_c with the minimum curvature value in $\mathbb{A}$ to start the expanding process to determine the local region of that point.

The chosen point, $\mathbf{p}_c$ will then be added to the local region set, $\mathbb{R}_l$ and removed from $\mathbb{A}$.

$$\mathbb{R}_l = \mathbb{R}_l \cup \{\mathbf{p}_c\} \tag{3}$$

$$\mathbb{A} = \mathbb{A} \setminus \{\mathbf{p}_c\} \tag{4}$$

Let $\mathbb{N}$ be the set of neighbours around $\mathbf{p}_c$. Each point, $\mathbf{p}_n \in \mathbb{N}$, is checked to see if it satisfies a specific angle threshold, θ_{th} and curvature threshold, c_{th}. A qualifying point set, $\mathbb{Q}$ will be added to $\mathbb{R}_l$ and removed from $\mathbb{A}$.

$$\mathbb{Q} = \{\mathbf{p_n} \in \mathbb{N} \mid \theta_{th} \geq acos(\vec{\boldsymbol{n}}_n.\vec{\boldsymbol{n}}_c),\ c_{th} \geq c_n - c_c\} \tag{5}$$

$$\mathbb{R}_l = \mathbb{R}_l \cup \mathbb{Q} \tag{6}$$

$$\mathbb{A} = \mathbb{A} \setminus \mathbb{Q} \tag{7}$$

where $\vec{\boldsymbol{n}}_n$ and c_n, $\vec{\boldsymbol{n}}_c$ and c_c are the normal vector and curvature values of points $\mathbf{p}_n$ and $\mathbf{p}_c$, respectively.

After all neighbours of $\mathbf{p}_c$ have been considered, $\mathbb{R}_l$ will be added to $\mathbb{R}$. Then the algorithm will choose another point in $\mathbb{A}$ and start the expanding process again until $\mathbb{A} = \varnothing$.

$$\mathbb{R} = \mathbb{R} \cup \mathbb{R}_l \tag{8}$$

2.2 Principal Component Analysis

Principal component analysis (PCA) is a multivariate technique with the goal being to extract and present important data as principal components [Abdi and Williams, 2010]. Initially, PCA was designed for digital image processing but was later extended to be implemented for point cloud data [Furferi *et al.*, 2011].

In a point cloud, P with N points, each point, $\mathbf{p}_i$ is presented as $[x_i, y_i, z_i]^T$ in the three-dimensional coordinate system. P can be represented as a N columns matrix:

$$\begin{aligned} \mathrm{P} &= [\,\mathbf{p}_1,\, \mathbf{p}_2,\, ...,\, \mathbf{p}_i,\, ...,\, \mathbf{p}_N\,] \in \mathbb{R}^{3\times \mathrm{N}} \\ \mathbf{p}_i &= [\,x_i, y_i, z_i\,]^T \in \mathbb{R}^{3\times 1}, \\ & i \in \mathbb{Z}, N \geq i \geq 0 \end{aligned} \tag{9}$$

The empirical mean, $\bar{\mathbf{m}} = [x_m, y_m, z_m]^T$ of P can be calculated as,

$$\bar{\mathbf{m}} = \frac{1}{N}\sum_{i=1}^{N}[x_i, y_i, z_i]^T \tag{10}$$

It is then possible to obtain a zero-centered deviation matrix, $\mathrm{D} \in \mathbb{R}^{3\times \mathrm{N}}$ by subtracting $\bar{\mathbf{m}}$ from every point in P,

$$\mathrm{D} = \mathrm{P} - \bar{\mathbf{m}}\,\mathrm{I}_{3\times \mathrm{N}} \tag{11}$$

The co-variance matrix, $\mathrm{C} \in \mathbb{R}^{3\times 3}$ can then be calculate as,

$$\mathrm{C} = \frac{1}{N-1}\mathrm{D}\mathrm{D}^T \tag{12}$$

C is now symmetrical, and by applying eigen decomposition the eigenvalues and eigenvectors of C are calculated,

$$\begin{aligned} &\mathrm{C} = \mathrm{V}\Lambda\mathrm{V}^T, \\ &\Lambda = \mathrm{diag}(\,\lambda_1, \lambda_2, \lambda_3\,),\ \lambda_1 > \lambda_2 > \lambda_3, \\ &\mathrm{V}\,\mathrm{V}^T = \mathrm{I}_{3\times 3} \end{aligned} \tag{13}$$

where Λ is a diagonal matrix of dimension 3×3, with the eigenvalues along the diagonal sorted in descending order; and V is a unitary matrix whose columns are the eigenvectors corresponding to the eigenvalues.

This operation is similar to performing Singular Value Decomposition (SVD) on D as,

$$\begin{aligned} &\mathrm{D} = \mathrm{U}\Lambda^{\frac{1}{2}}\mathrm{V}^T, \\ &\Lambda^{\frac{1}{2}} = \mathrm{diag}(\,\lambda_1^{\frac{1}{2}}, \lambda_2^{\frac{1}{2}}, \lambda_3^{\frac{1}{2}}\,) \end{aligned} \tag{14}$$

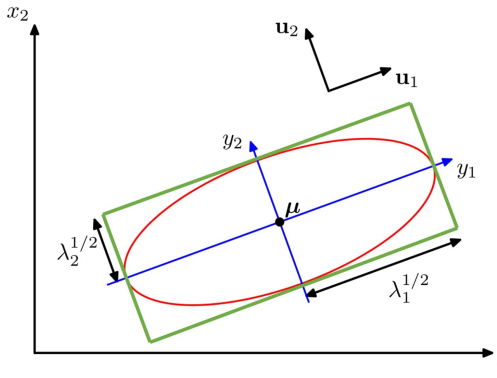


Figure 4: PCA reveals the natural distribution of normally distributed data [Bishop, 2006]. Here eigenvectors are labeled as $\mathbf{u}_1$ and $\mathbf{u}_2$. The data mean is located at the center of the surface. Standard deviations along each axis are the singular values $\lambda_1^{\frac{1}{2}}$ and $\lambda_2^{\frac{1}{2}}$. The ratio of length-to-height is $\lambda_1^{\frac{1}{2}} : \lambda_2^{\frac{1}{2}}$.

The singular vectors and eigenvectors from both operations are the same. The difference is that the singular values from SVD are square roots of the corresponding eigenvalues. This is since the eigenvectors represent the underlying orthogonal axis of uncorrelated data distribution. For a symmetrical geometrical shape, the eigenvectors give the right choice of shape axis, and the empirical mean coincides with the shape's centroid. In a cloud patch, the PCA axis should be aligned with the rectangle's natural shape axis, and the PCA mean is at the rectangle's center. This concept is illustrated in Figure 4.

Moreover, the singular values are the standard variation of data distributions along each latent axis. The singular value ratios should be indicative of the length-to-height ratio of the rectangular brick surface, with each class of brick having a unique length-to-height ratio (see Table 4 for details). It is therefore possible to classify the brick type by computing this singular value ratio, or eigenvalue ratio. Performing eigen decomposition on a symmetric matrix is much cheaper than SVD. Thus, PCA is selected instead of direct SVD.

2.3 Proposed Framework

Preliminary Processing

Firstly consider a fixed RGB-D camera, a robotic arm and the relative location of the camera in the robot's frame. Data collected from the camera is a colour point cloud and all objects in the field of view are immobile. Let the robotic arm's base be the global frame, the camera's position and orientation be represented by a 3×1 translation vector, $\mathbf{t}$, and a 3×3 rotation matrix, R:

$$\mathbf{t} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}, \mathrm{R} = \mathrm{R}_z(\gamma)\mathrm{R}_y(\beta)\mathrm{R}_x(\alpha) \tag{15}$$

where R_x, R_y, R_z are the rotation matrices about axes x, y, z for roll, pitch, yaw angles α, β, γ, respectively.

Note that the point cloud data, P_c coming directly from the RGB-D camera is noisy and contains a large amount of information to be processed. In this paper, a few layers of filters are implemented to improve the quality of the received point clouds. Initially, the data from the camera is run through a voxel grid filter to reduce the number of points to a preferred resolution. Voxel grid filtering also guarantees points in the cloud are evenly distributed.

The down-sampled point cloud, P_d is then filtered by the Z-axis value of each point. Since noisy data cannot be processed and points far away from the camera are generally noisier, all $\mathbf{p}_i$ that exceed a distance limit, z_{limit} will be removed. Then,

$$\mathrm{P}_f^d = \{\mathrm{P}_d \setminus \mathbf{p}_i \mid \mathbf{p}_i \in \mathrm{P}_d, z_i \geq z_{limit}\}, \tag{16}$$

where P_f^d is the point cloud after filtering by distance. This point cloud is further filtered by the colour of the object of interest. This filter layer's purpose is not to recognise the object but to limit the range of data to be considered for later steps. Therefore, the colour range does not have to be strict. The resulting cloud, P_f^c can be defined as,

$$\mathrm{P}_f^c = \{\mathrm{P}_f^d \setminus \mathbf{p}_i \mid \mathbf{p}_i \in \mathrm{P}_f^d,\ h_{lowlim} \geq h_i \geq h_{hilim}\} \tag{17}$$

where the considering point colour is described in HSV colour space and h_{hilim} and h_{lowlim} are the high limit and low limit of the "Hue" value of the point.

Region Growing Segmentation

Region growing segmentation is then implemented on P_f^c. Region growing segmentation required each point in P_f^c to have a corresponding normal vector. These normal vectors are computed for each point by computing the co-variance matrix of the query point and its neighbours. The C of $\mathbf{p}_i$ can be obtain using the following equation,

$$\mathrm{C} = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{p}_i - \bar{\mathbf{p}})(\mathrm{p}_i - \bar{\mathbf{p}})^T,\ \mathrm{C}\vec{\boldsymbol{v}}_{\boldsymbol{j}} = \lambda_j \vec{\boldsymbol{v}}_{\boldsymbol{j}},\ j \in \{0,1,2\} \tag{18}$$

where k is the number of neighbor points considered of $\mathbf{p}_i$, $\bar{\mathbf{p}}$ is the centroid of $\mathbf{p}_i$ and its neighbours, and λ_j and $\vec{\boldsymbol{v}}_j$ are the j-th eigenvalue and eigenvector of C.

Object Recognition And Pose Estimation

Region growing segmentation results is a set of clusters, $\mathbb{C}_{rg}$. Each element in the cluster is a point considered to be on the same surface. The result from region growing segmentation is further filtered by the number of points, N_{th} in clusters. This step is designed to removed clusters that do not have enough points to produce an accurate final result.

$$\mathbb{C}_{rg}^f = \mathbb{C}_{rg} \setminus \mathbb{C}_{uq}, \tag{19}$$

where $\mathbb{C}_{rg}^f$ is the set of qualified clusters and $\mathbb{C}_{uq}$ is a set of clusters with less than N_{th} points.

For surface, S_i from $\mathbb{C}_{rg}^f$, PCA is applied to obtain the principal axes X, Y and Z and the centroid.

It is necessary that the unit vector of the Z-axis, $\vec{\boldsymbol{u}}_i^z$ of the obtained cloud on S_i, points away from the camera's viewpoint. This requirement is checked by calculating the dot product, d_i of $\vec{\boldsymbol{u}}_i^z$ and the unit vector of the Z-axis of the camera, $\vec{\boldsymbol{u}}_{cam}^z$. Since all calculations so far are in the camera frame, $\vec{\boldsymbol{u}}_{cam}^z = (0,0,1)$. If $d_i \geq 0$ then $\vec{\boldsymbol{u}}_i^z$ is pointing away from the camera. However, if $0 \geq d_i$ then $\vec{\boldsymbol{u}}_i^z$ is pointing towards the camera and the coordinate system obtained from PCA will be turn by π around the X-axis.

The geometric attributes of S_i are considered in the rectangular object recognition task. A comparison between the known model edges length with surface S_i edges is executed to determine if S_i is a surface of the targeted rectangular prism-shaped object. The edge length calculations of S_i are based on the maximum distance, d_x^{max} and d_y^{max} from any point on S_i to the X and Y axes at the centroid of that surface.

The width, w and height, h of the surface will be twice the value of d_x^{max} and d_y^{max}. If the calculated values are similar to the model, S_i is a surface of the targeted rectangular object.

$$w = 2 \times d_x^{max},\ h = 2 \times d_y^{max} \tag{20}$$

Surface Selection

If the S_i is a surface of the targeted rectangular prism-shaped object, it will then be stored in a set, $\mathbb{S}_r$ for the best surface selection process.

$$\mathbb{S}_r = \mathbb{S}_r \cup \{S_i\} \tag{21}$$

Each surface S_i of $\mathbb{S}_r$ contains a centroid point, $\mathbf{c}_i$ and unit vectors of that surface's coordinate system. For best surface selection the following parameters are considered: the distance of the centroid to the global frame or the robotic arm's base, d_i^g, the angle between S_i and the global frame, θ_i^g, and the number of points, N_i belong to the surface. Each of these parameters are calculated and used as a metric in surface candidate selection. To combine all tests fairly with equal voting power, each raw value is mapped to the same score range [0..1] using the Sigmoid function [Kros *et al.*, 2006] as,

$$\begin{aligned} s_i^d &= \frac{2}{1+\exp\frac{6(0.3-\text{distance})}{3}} - 1 \\ s_i^\theta &= \frac{2}{1+\exp\frac{6(0.0-\text{angle})}{1.0}} - 1 \\ s_i^N &= \frac{2}{1+\exp\frac{6(1000-\text{ptcount})}{20000}} - 1 \end{aligned} \tag{22}$$

where s_i^d is the score for distance, s_i^θ is the score for the angle, and s_i^N for number of points in region. The final score, s_i^f is the product of all three scores,

$$s_i^f = s_i^d * s_i^\theta * s_i^N \tag{23}$$

The product rule here is to ensure the candidate with a poor score in either criterion will receive the highest penalty. The surface with the highest score will be the best surface with well-balanced attributes. This surface is selected as the final output for subsequent processing.

2.4 Rotation Averaging

The robot is stationary while capturing depth data from the camera. The best surface that is selected will always be the same for multiple runs, $[1..N_T]$ in this time interval. To further improve the result, an averaging calculation is applied over the calculated poses. The results of each run will be collected in $\mathbb{S}_T$. The average calculation will be executed for all elements of $\mathbb{S}_T$.

There are two components needed to be averaged: the centroid position and the orientation. Each surface, $S_t \in \mathbb{S}_T$ has a centroid, $\mathbf{c}_t$ and its orientation presented by the coordinate system with unit vectors: $\vec{\boldsymbol{u}}_t^x$, $\vec{\boldsymbol{u}}_t^y$, and $\vec{\boldsymbol{u}}_t^z$. The averaged centroid position $\mathbf{c}_{avg}$ will be calculated as,

$$\mathbf{c}_{avg} = \frac{1}{N_T}\sum_{t=1}^{N_T}\mathbf{c}_t \tag{24}$$

where N_T is the number of elements in the set $\mathbb{S}_T$.

The best orientation, R_{avg} from all results are achieved by solving the single rotation averaging problem as described in [Hartley *et al.*, 2013]. During single rotation averaging, several rotation estimates of a still target are computed over time and then being averaged to give the best result [Hartley *et al.*, 2013], thereby reducing output pose noise. To compute the optimal rotation, a rotation difference metric is defined according to [Hartley *et al.*, 2013]

$$\begin{aligned} \text{Let} \quad & d_{chord}(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} \cdot \mathbf{q}^{-1} - \mathbf{e}\| \\ \text{where,} \quad & \mathbf{p} \cdot \mathbf{q}^{-1} \in \mathbb{R}^{4\times 1}, \quad \mathbf{e} = (1,0,0,0), \\ \text{define} \quad & \mathbf{p} \cdot \mathbf{q}^{-1} = (\cos(\psi), \sin(\psi)\mathbf{u}) \\ \text{therefore:} \quad & \|\mathbf{p} \cdot \mathbf{q}^{-1} - \mathbf{e}\| = 2\sin(\psi/2). \end{aligned} \tag{25}$$

This derivation suggests the distance between two unit 4-vector, also known as the chordal distance is directly related to their separation angle, ψ. [Hartley *et al.*, 2013] then uses the chordal distance as the error metric to define a cost function, $C(\mathrm{R}_{avg})$ under L_2 chordal distance as,

$$\begin{aligned} C(\mathrm{R}_{avg}) &= \sum_{t=1}^{N_T} d_{\mathrm{chord}}(\mathrm{R}_i,\ \mathrm{R}_{avg})^2 \\ &= \sum_{t=1}^{N_T} d_{\mathrm{chord}}(\mathbf{q}_i,\ \mathbf{q}_{avg})^2 \end{aligned} \tag{26}$$

where R_t is a rotation of a surface $S_t \in \mathbb{S}_T$ and $\mathbf{q}_t$ is R_t's $\mathbb{SO}(3)$ equivalent in unit quaternion form.

The rotation averaging problem is then to find R_{avg} such that $C(\mathrm{R})$ is minimised. To this end, the rotation matrix $\mathrm{R}_t \in \mathbb{R}^{3\times 3}$ is converted to the unit quaternion form: $\boldsymbol{q}_t = [q_w, q_x, q_y, q_z]^T \in \mathbb{R}^{4\times 1}$ and $\|\boldsymbol{q}_t\| = 1$.

$$\begin{aligned} q_w &= \sqrt{1 + \mathrm{R}_{00} + \mathrm{R}_{11} + \mathrm{R}_{22}}\,/\,2 \\ q_x &= (\mathrm{R}_{21} - \mathrm{R}_{12})\,/\,(4\,q_w) \\ q_y &= (\mathrm{R}_{02} - \mathrm{R}_{20})\,/\,(4\,q_w) \\ q_z &= (\mathrm{R}_{10} - \mathrm{R}_{01})\,/\,(4\,q_w) \end{aligned} \tag{27}$$

In the next step, the matrix, $\mathrm{A} \in \mathbb{R}^{4\times 4}$ is constructed as the sum of dyadic product of $\boldsymbol{q_t}\boldsymbol{q_t}^T$ over time.

$$\mathrm{A} = \sum_{t=1}^{N_T} \boldsymbol{q_t}\boldsymbol{q_t}^T \tag{28}$$

Hartley et al., [Hartley *et al.*, 2013] provided proof that the eigenvector, $\boldsymbol{s}_{max}$ of matrix, A corresponding to its largest eigenvalue minimises the cost function in (26). Thus, $\boldsymbol{s}_{max}$ is the averaged rotation in the optimal sense.

3 Experiments

Two experiments were conducted to display the result of the proposed pipeline. The first compared the pose estimated from the proposed framework with the well-known ICP approach [Simon *et al.*, 1994]. The results are evaluated based on the ground truth data. The second experiment tested the ratio between eigenvalues and the size on multiple rectangular objects with different dimensions.

All experiments were conducted in the Gazebo simulation environment on a machine with an Intel Core i7-7700HQ CPU, 16 GB RAM, and Geforce GTX 1050 GPU.

3.1 Comparative Study: Proposed Framework and ICP

The simulation consists of a mobile base with a simulated model of the Realsense D435 attached. The mobile base will be placed in front of a pile of green bricks, which are a class of brick with the same dimensions: $0.6 \times 0.2 \times 0.2$ meters. Point cloud data is obtained directly from the camera's point cloud topic. Figure 5 shows five different scenario of the experiment. For each scenario, the proposed pipeline will be implemented to obtain the best surface for the picking task. ICP will then try to fit this surface to a generated model. This model is a cloud of a surface with the same colour and size as the brick. The model contains points evenly distributed, similar to the result after the voxel grid filter process. The results from ICP and the proposed framework are compared to the ground truth value. Figure 6 shows the segmented point cloud in each scenario. The bricks on which the selected surfaces belong to can be observed. Then, the ground truth value of the surface can be determined from the Gazebo world model.

Each method result is evaluated by the distance error and angular error. The distance error, e_d is calculated by the distance between the result's centroid with the ground truth data.

$$e_d = ||\mathbf{c}_t - \mathbf{c}_g|| \tag{29}$$

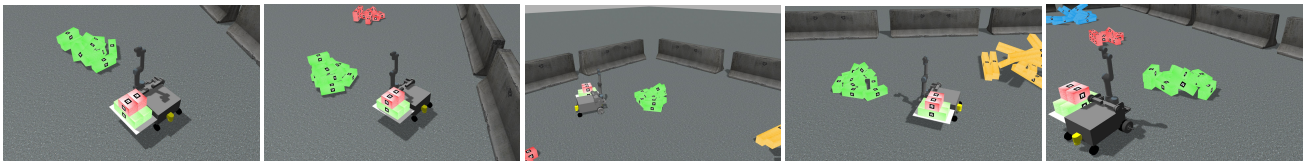
where $\mathbf{c}_t$ is the centroid point position from either of the two methods, and $\mathbf{c}_g$ is the ground truth value.

The angular error, e_{th} is calculated by the difference between the Z-axis obtained from the result and the ground truth Z-axis.

$$e_{th} = cos^{-1}(\vec{\boldsymbol{u}}_r^z \cdot \vec{\boldsymbol{u}}_g^z) \tag{30}$$

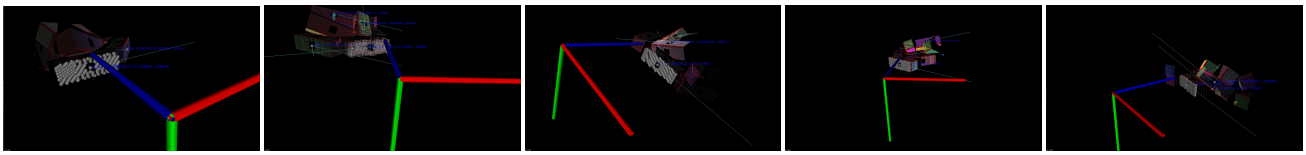
where $\vec{\boldsymbol{u}}_r^z$ and $\vec{\boldsymbol{u}}_g^z$ are the unit vectors of the Z axes obtained from roll, pitch and yaw values from the methods and the ground truth data. The unit vector of Z-axis can be calculated as,

$$\vec{\boldsymbol{u}}^z = \begin{pmatrix} \cos\alpha \sin\beta \cos\gamma + \sin\alpha \sin\gamma \\ \sin\alpha \sin\beta \cos\gamma - \cos\alpha \sin\gamma \\ \cos\beta \cos\gamma \end{pmatrix} \tag{31}$$



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4 (e) Scenario 5

Figure 5: Experiment 1 scenarios.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4 (e) Scenario 5

Figure 6: Segmented point cloud results from Experiment 1.

The comparative centroid estimation results are recorded in Table 1. Table 2 contains the estimated orientation and the calculated orientation error. Table 3 displays the execution time of each method in milliseconds and the ratio between the two methods.

Table 1: A comparison of the centroid estimation accuracy between ICP and our method.

Ground Truth Centroid			Output Centroid						Error	
			ICP			Ours			ICP	Ours
x	y	z	x	y	z	x	y	z		
-5.3	5.5	0.1	-5.4	5.5	0.1	-5.4	5.5	0.1	0.05	**0.05**
-5.7	4.9	0.1	-5.9	4.6	0.1	-5.7	4.9	0.1	0.31	**0.01**
-5.2	4.3	0.1	-5.7	4.5	0.1	-5.2	4.3	0.1	0.31	**0.04**
-4.6	4.2	0.1	-4.5	4.3	0.1	-4.5	4.2	0.1	0.13	**0.06**
-4.6	4.8	0.1	-4.5	4.7	0.1	-4.6	4.8	0.1	0.08	**0.06**

Table 2: A comparison of the orientation estimation accuracy between ICP and our method.

Ground Truth Orientation			Output Orientation						Error	
			ICP			Ours			ICP	Ours
α	β	γ	α	β	γ	α	β	γ		
1.6	0.0	0.5	-1.6	0.0	-2.7	1.6	0.0	0.5	0.06	**0.00**
0	0	1.7	-1.6	-0.0	-1.7	1.6	0.0	1.5	0.16	**0.02**
-1.6	0.0	2.8	-1.6	0.0	-0.3	-1.6	0.0	-0.3	0.06	**0.06**
0.0	0.0	0.6	-1.6	-0.0	-2.6	-1.6	0.0	0.6	0.06	**0.01**
3.1	0.0	-1.5	-1.6	-0.0	1.7	-1.6	-0.0	1.7	0.06	**0.06**

3.2 Ratio Between Eigenvalue and Object Dimensions

This experiment evaluates the relationship among the eigenvalues ratios and the dimension ratio for each type of brick. The simulation environment for this experiment is similar to the previous experiment, with two extra piles of blue and orange bricks added. The newly added blue bricks and orange bricks' dimensions are $1.2 \times 0.2 \times 0.2$ meters and $1.8 \times 0.2 \times 0.2$ meters, respectively. The mobile base is placed at multiple points of view for each pile. PCA is then executed on the point cloud that is recorded from each perspective. The two larger eigenvalues, corresponding to the length and height of a surface, are considered. Each brick colour has a different length-to-height ratio. Therefore, the ratio of the two larger eigenvalues obtained from PCA are expected to change between different bricks exponentially.

The experiment data are recorded in Table 4.

Table 3: A comparison of execution time between ICP and our method.

ICP	Ours	Ratio
33	496	15.03
48	485	10.10
41	434	10.59
53	338	6.38
33	365	11.07

4 Discussion

The first experiment demonstrated the ability of the proposed method to precisely estimated the pose of a single rectangular surface within a pile of objects with similar colour and dimensions. In comparison with using ICP for pose estimation, the results from the proposed framework are superior for both locating the centroid point and estimating the surface's orientation.

Table 4: A comparison of the brick's PCA ratio with size ratio. $(\frac{L}{H})^2$ refers to the length-to-height ratio squared, and λ_i refers to the i'th eigenvalue from PCA.

Brick colour	Expected dimension	Expected $(\frac{L}{H})^2$	Test$^{(1)}$: λ_1/λ_2 Test$^{(2)}$: λ_1/λ_2 Test$^{(3)}$: λ_1/λ_2	$= \lambda^{(1)}_{ratio}$ $= \lambda^{(2)}_{ratio}$ $= \lambda^{(3)}_{ratio}$
green -brick	length: 0.6 width: 0.2 height: 0.2	9 : 1	155.8 / 16.4 84.3 / 8.86 103.5 / 12.2	= **9.5** = **9.5** = **8.5**
blue -brick	length: 1.2 width: 0.2 height: 0.2	36 : 1	1223.7 / 31.8 1042.6 /30.7 1215.6/ 32.1	= **38.5** = **34.0** = **37.8**
orange -brick	length: 1.8 width: 0.2 height: 0.2	81 : 1	4033.6 / 47.9 4016.8 / 47.8 1637.6 / 20.2	= **84.2** = **84.0** = **80.9**

Table 1 shows the distance error between the ground truth value and the centroid points calculated by either ICP or the proposed framework. It can be observed that the results from the proposed framework are up to 30 times better in terms of distance error. Furthermore, the distance error produced from the proposed method are less than 0.1 meters, while that of ICP has a higher chance (60%) to exceed 0.1 meters. The suggested pipeline surpasses the ICP method because the suggested approach takes into consideration the effect of noise on point cloud data, while ICP did not.

Table 2 shows the rotation error between the normal vectors of the ground truth plane and the Z-axis produced by either ICP or the presented approach. The rotation estimated by both ICP and the proposed approach has a high precision. However, implementing ICP on a single surface results in multiple solutions. ICP is able to find more than one possible solution since there are no conditions set for the rotation obtained by ICP. Therefore, the ICP results are inconsistent. Due to this inconsistency, rotation averaging cannot be applied. During experiments, there is one scenario where the ICP result is offset by π, which violates the requirement of the Chordal L_2-Mean, i.e., the difference between two input rotations must not exceed $\pi/4$.

The second experiment displays the relationship between calculated eigenvalues ratio of PCA and length-to-height ratio of the surface. Table 4 shows the calculated eigenvalues ratio are within 10% of the expected value over 9 tests on three different models.

Table 3 demonstrates the trade-off between accuracy and processing time. Despite the proposed framework's high accuracy and robustness as discussed above, it requires significantly more time to execute compared to ICP.

5 Conclusions

This paper has presented an approach for the problem of rectangular prism-shaped object recognition and precise pose estimation. Experiments conducted have shown that this approach is successful in obtaining the pose of a single object within a set of multiple similar objects. Experiments also proved the framework introduced in this paper is more accurate and consistent compared to the Iterative Closest Point (ICP) approach. Furthermore, since the proposed approach is designed for open-loop systems, internal procedures can be easily modified without damaging the overall framework.

Potential extensions to this approach include testing with multiple models, and having different dimensions and surface texture to determine the viability and robustness of the presented pipeline. Other criteria will also be considered to further improve the accuracy and efficiency.

References

[Abdi and Williams, 2010] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[Abraham and Inouye, 2014] Gad Abraham and Michael Inouye. Fast principal component analysis of large-scale genome-wide data. *PloS one*, 9(4):e93766, 2014.

[Adams and Bischof, 1994] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6):641–647, 1994.

[Bishop, 2006] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[Brown and Asada, 2007] Christopher Y Brown and H Harry Asada. Inter-finger coordination and postural synergies in robot hands via mechanical implementation of principal components analysis. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2877–2882. IEEE, 2007.

[Chatterjee and Madhav Govindu, 2013] Avishek Chatterjee and Venu Madhav Govindu. Efficient and robust large-scale rotation averaging. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 521–528, 2013.

[Furferi *et al.*, 2011] Rocco Furferi, Lapo Governi, Matteo Palai, and Yary Volpe. From unordered point cloud to weighted b-spline-a novel pca-based method. In *Applications of Mathematics and Computer Engineering-American Conference on Applied*

Mathematics, AMERICAN-MATH, volume 11, pages 146–151, 2011.

[Gamage and Lasenby, 2002] Sahan S Hiniduma Udugama Gamage and Joan Lasenby. New least squares solutions for estimating the average centre of rotation and the axis of rotation. *Journal of biomechanics*, 35(1):87–93, 2002.

[Hartley *et al.*, 2013] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International journal of computer vision*, 103(3):267–305, 2013.

[He *et al.*, 1995] Taosong He, Lichan Hong, Arie Kaufman, Amitabh Varshney, and Sidney Wang. Voxel based object simplification. In *Proceedings of the 6th conference on Visualization'95*, page 296. IEEE Computer Society, 1995.

[Kros *et al.*, 2006] John F Kros, Mike Lin, and Marvin L Brown. Effects of the neural network s-sigmoid function on kdd in the presence of imprecise data. *Computers & operations research*, 33(11):3136–3149, 2006.

[Kuncheva and Faithfull, 2013] Ludmila I Kuncheva and William J Faithfull. Pca feature extraction for change detection in multidimensional unlabeled data. *IEEE transactions on neural networks and learning systems*, 25(1):69–80, 2013.

[Li and Chutatape, 2004] Huiqi Li and Opas Chutatape. Automated feature extraction in color retinal images by a model based approach. *IEEE Transactions on biomedical engineering*, 51(2):246–254, 2004.

[Paul *et al.*, 2013] Gavin Paul, Ngaiming Kwok, and Dikai Liu. A novel surface segmentation approach for robotic manipulator-based maintenance operation planning. *Automation in Construction*, 29:136–147, 2013.

[Paul *et al.*, 2016a] Gavin Paul, LiYang Liu, and Dikai Liu. A novel approach to steel rivet detection in poorly illuminated steel structural environments. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–7. IEEE, 2016.

[Paul *et al.*, 2016b] Gavin Paul, Shuyuan Mao, Liyang Liu, and Rong Xiong. Mapping repetitive structural tunnel environments for a biologically-inspired climbing robot. In *ASSISTIVE ROBOTICS: Proceedings of the 18th International Conference on CLAWAR 2015*, pages 325–333. World Scientific, 2016.

[Simon *et al.*, 1994] David A Simon, Martial Hebert, and Takeo Kanade. Real-time 3-d pose estimation using a high-speed range sensor. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 2235–2241. IEEE, 1994.

[Sol, 2017] Joan Sol. Quaternion kinematics for the error-state kalman filter, 2017.

[Sophian *et al.*, 2003] Ali Sophian, Gui Yun Tian, David Taylor, and John Rudlin. A feature extraction technique based on principal component analysis for pulsed eddy current ndt. *NDT & e International*, 36(1):37–41, 2003.

[Vo *et al.*, 2015] Anh-Vu Vo, Linh Truong-Hong, Debra F Laefer, and Michela Bertolotto. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104:88–100, 2015.

[Wang, 1969] James C Wang. Variation of the average rotation angle of the dna helix and the superhelical turns of covalently closed cyclic λ dna. *Journal of molecular biology*, 43(1):25–39, 1969.

[Yang *et al.*, 2004] Jian Yang, David Zhang, Alejandro F Frangi, and Jing-yu Yang. Two-dimensional pca: a new approach to appearance-based face representation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 26(1):131–137, 2004.