Supplementary Information for
Primitive Shape Recognition for Object Grasping

Yunzhi Lin, Chao Tang, Fu-Jen Chu, Ruinian Xu, and Patricio A. Vela

I. Shape Fitting

The shape fitting algorithm applied as part of the PS-CNN pipeline follows the steps shown in Algorithm 1. The steps are applied for each segmented primitive shape region. Step 1 extracts the point cloud in the camera frame $\mathcal{C}$ from the shape region, the depth image, then maps it to the world frame $\mathcal{W}$ using the $SE(3)$ element describing the camera frame relative to the world frame, $g_{\mathcal{C}}^{\mathcal{W}}$. Step 2 identifies a region of interest (ROI) $\mathcal{R}$ on the (tabletop) support surface, which is then used to remove outliers near the surface. Step 3 denoises the point cloud based on a Euclidean distance agglomerative clustering method. Step 4 uses PCA to obtain the reference vector V of the object if V is not provided. Step 5 employs a RANSAC-based approach to fit the primitive shapes, where three different strategies correspond to cuboid-like, cylinder-like and sphere-like objects, respectively.

The following subsections discuss the RANSAC algorithms for these shape classes. The general process involves generating samples and model estimates, model confirmation, model scoring, threshold updating, and model selection. Model confirmation is an additional step that serves to reject models with poor conditioning, inconsistent geometry relative to given prior, or whose parameters lie outside of established intervals. Once the iterative process stops, the best of the stored models is chosen as the final shape fit.

Algorithm 1 Shape Fitting Framework

Require: a depth map I; a segmentation mask M; transformation matrix from the camera frame to the world frame $g_{\mathcal{C}}^{\mathcal{W}}$; maximum iteration count N; shape class C; angular distance threshold $d_{\angle}$.
Optional: reference vector V.
Ensure: a shape model x.
1: $\mathcal{P} \leftarrow$ PreprocessCloud$(I, M, g_{\mathcal{C}}^{\mathcal{W}})$
2: $\mathcal{P} \leftarrow$ GetROI$(\mathcal{P})$
3: $\mathcal{P} \leftarrow$ Denoise$(\mathcal{P})$
4: $V \leftarrow$ PCA$(\mathcal{P})$ if $V = \varnothing$
5: $x \leftarrow$ RANSAC$(\mathcal{P}, N, \mathrm{C}, V, d_{\angle})$

1) RANSAC-based Cylinder Shape Fitting : Algorithm 2 describes the shape fitting algorithm for cylinder-like objects. The basic idea is to generate shape model candidates in large quantities and maximize the evaluated score based on inliers [1]. To begin requires establishing a maximum candidate count $\mathcal{N}_x$ and a score threshold $\bar{\sigma}$ based on the shape class C and the number of points in the point cloud $|\mathcal{P}|$ (Step 2). In the RANSAC loop, a shape model candidate x is generated from $\mathcal{P}$ [1] (Step 4). Next, the angular difference between the candidate's shape axis and the reference vector V is compared to the angular distance threshold $d_{\angle}$ (Step 5). If the check passes the threshold, then the model is provisionally accepted and scored against the point cloud.

Algorithm 2 RANSAC-based Cylinder Shape Fitting

Require: a point cloud $\mathcal{P}$; maximum iteration count N; shape class C; reference vector V; angular distance threshold $d_{\angle}$.
Ensure: a shape model x.
1: $i = 0$, $n_x = 0$, $\sigma_{\text{list}} = \varnothing$
2: $(\mathcal{N}_x, \bar{\sigma}) =$ initParam$(\mathrm{C}, |\mathcal{P}|)$
3: **while** $i < N$ **and** $n_x < \mathcal{N}_x$ **do**
4: $x =$ genCylinder$(\mathcal{P})$
5: **if not** checkAxis$(x; V, d_{\angle})$ **then**
6: **continue**
7: **end if**
8: $\hat{\sigma} =$ scoreCylinder$(x; \mathcal{P})$
9: $\bar{\sigma} =$ updateThresh$(i, n_x, \mathrm{C}, |\mathcal{P}|)$
10: **if** $\sigma^1 > \bar{\sigma}$ **then**
11: $\sigma_{\text{list}} = \sigma_{\text{list}} \cup \hat{\sigma}$
12: $\mathcal{X} = \mathcal{X} \cup x$
13: $n_x = n_x + 1$
14: **end if**
15: $i = i + 1$
16: **end while**
17: $x =$ selectScore$(\mathcal{X}; \sigma_{\text{list}}, \mathrm{C}, |\mathcal{P}|)$

Scoring relies on the Euclidean distances of points in the point cloud to the cylinder (based on its shape axis and radius). Points with distance values lying within a range determined by the shape class C will be deemed inliers and added to the inlier set $\mathcal{I}_{\text{in}}$. The three scores computed are: 1. the number of inliers $|\mathcal{I}_{\text{in}}|$; 2. the ratio of the number of inliers $|\mathcal{I}_{\text{in}}|$ to the number of points in the point cloud $|\mathcal{P}|$; 3. the number of inliers per unit area on the shape surface (Step 8). The score threshold $\bar{\sigma}$ is updated according to the iteration number i, the candidate number n_x, the shape class C, and the number of points in the point cloud $|\mathcal{P}|$. This design speeds up the candidate generation process when there are few valid candidates generated relative to the loop iterations (Step 9). The reduced threshold becomes more permissive regarding model acceptance. If the first type of score σ^1 is above the score threshold $\bar{\sigma}$, $\hat{\sigma}$ and x will be saved in the score list σ_{list} and the model set $\mathcal{X}$, respectively

Algorithm 3 RANSAC-based Cuboid Shape Fitting

Require: a point cloud $\mathcal{P}$; maximum iteration count N; shape class C; reference vector V; angular distance threshold $d_\angle$.
Ensure: a shape model x.

1: $i = 0$, $\sigma_{\text{best}} = 0$, $\mathcal{B} = \varnothing$.
2: $\bar{\sigma} = \text{initParam}(|\mathcal{P}|)$
3: **while** $i < N$ **and** $\sigma_{\text{best}} < \bar{\sigma}$ **do**
4: $(\mathcal{X}, \mathcal{B}) = \text{genCuboid}(\mathcal{P}; i, \mathcal{B})$ (see Algorithm 4)
5: **for** all $x \in \mathcal{X}$ **do**
6: **if not** checkLength(x) **then**
7: **continue**
8: **end if**
9: **if not** checkAxis$(x; V, d_\angle)$ **then**
10: **continue**
11: **end if**
12: $\sigma = \text{scoreCuboid}(x; \mathcal{P})$ (see Algorithm 5)
13: $\bar{\sigma} = \text{updateThresh}(\bar{\sigma}, \mathcal{B}, |\mathcal{P}|)$
14: **if** $\sigma > \sigma_{\text{best}}$ **then**
15: $\sigma_{\text{best}} = \sigma$
16: $x_{\text{best}} = x$
17: **end if**
18: **end for**
19: $i = i + 1$
20: **end while**
21: $x \leftarrow x_{\text{best}}$

(Step 12). After finishing the RANSAC loop, the best model candidate x will be selected from the model set $\mathcal{X}$ based on its corresponding score σ saved in the score list σ_{list}. Which type of score to prioritize for selection is determined by the shape class C and the number of the points in the point cloud $|\mathcal{P}|$ (Step 17).

2) RANSAC-based Sphere Shape Fitting : The sphere-like shape fitting algorithm roughly follows the cylinder fitting process. The main differences is with the candidate generation function (Step 4) and the threshold setting (Step 9), which are adapted from [2] instead. The two scores computed are: 1. the number of inliers $|\mathcal{I}_{\text{in}}|$; 2. the number of inliers per unit volume (Step 8).

3) RANSAC-based Cuboid Shape Fitting: Algorithm 3 outlines the shape fitting algorithm for cuboid-like objects inspired by [3].

Main structure. The structure could be divided into 5 major parts: 1) Generation of cuboid candidates (Step 4, see Algorithm 4 for more detail); 2) Filtering process based on priors, where candidates which have a thin face will be filtered out to avoid overfitting (Step 6); 3) If given the reference vector V and the angular distance threshold $d_\angle$, the error between the model's shape axis and V has to be within $d_\angle$ (Step 9); 4) Score function (Step 12, see Algorithm 4 for more detail); 5) Threshold setting, where the score threshold $\bar{\sigma}$ is updated based on its previous value, the ratio record $\mathcal{B}$, and the number of points in the point cloud $|\mathcal{P}|$. It will be loosened to speed up the process if the record $\mathcal{B}$ shows that the previous strict threshold limits the speed of candidate generation (Step 13). In the end, a model candidate x will be returned corresponding to the best score σ_{best} (Step 21).

Algorithm 4 Cuboid Candidate Generation Function

Require: a point cloud $\mathcal{P}$; iteration times i; ratio record $\mathcal{B}$.
Ensure: a shape model set $\mathcal{X}$; ratio record $\mathcal{B}$.

1: $\mathcal{X} = \varnothing$
2: $\tau_r = \text{initParam}(|\mathcal{P}|)$
3: $P = \text{pointSelect}(\mathcal{P})$
4: $P_H = \text{pointConvexhull}(P)$
5: **for** all $P_h \subset P_H$ **do**
6: $(\mathcal{PL}^1, r^1) = \text{genPlane}(P_h)$
7: **if** $r^1 < \tau_r$ **then**
8: **continue**
9: **end if**
10: $P_C = \text{genCom}(P; P_h)$
11: **for** all $(p_1, p_2) \subset P_C$ **do**
12: $(\mathcal{PL}^2, r^2) = \text{genPlane}(\mathcal{PL}^1, p_1)$
13: **if not** checkRatio$(r^1, r^2; i, \mathcal{B})$ **then**
14: **continue**
15: **end if**
16: **if** checkPlane$(\mathcal{PL}^2; P, p_1, p_2)$ **then**
17: $\mathcal{PL}^{3-6} = \text{genPlane}(\mathcal{PL}^1, \mathcal{PL}^2; P)$
18: x = interPlane$(\mathcal{PL}^{1-6})$
19: $\mathcal{X} = \mathcal{X} \cup x$
20: $\mathcal{B} = \mathcal{B} \cup (r^1, r^2)$
21: **end if**
22: **end for**
23: **end for**

Generation of cuboid candidates. Algorithm 4 describes the cuboid model generation process. With 9 sampled points from the point cloud, it aims to identify 6 candidate faces defining a cuboid. The faces are determined one at a time and should maximize the number of inlier points when they are compared against the plane models for the faces. The final cuboid model is represented by the 8 intersection points of the 6 faces. Detail for the steps are given below.

First, 9 points are uniformly, randomly sampled from $\mathcal{P}$ to generate $P \subset \mathcal{P}$ (Step 3). Then a convex hull of P is computed and its vertices are extracted as $P_H \subset P$ (Step 4). The collection of planes as obtained from point triplets in the convex hull define the sources for building candidate sets of cuboid faces. Let a triplet define the $\mathcal{PL}^1$, which is potentially a cuboid face. From the plane, compute the r^1 based on the number of points near to $\mathcal{PL}^1$ (e.g., inlier points) relative to $|\mathcal{P}|$ (Step 6). The r^1 should exceed the ratio threshold τ_r in order to establish it as a dominant plane; it should fit more of the source points in $\mathcal{P}$ as compared to most other plane candidates (Step 7). Finding the dominant plane improves the chances of generating a valid cuboid shape model candidates in the subsequent steps. The plane normal is given by V_{N^1}.

Next, generate all possible point pairs P_C from the remaining 6 points $p \in (P \setminus P_h)$ (Step 10). Assume one of the points in the pair (p_1) lie on another face, then the face's plane normal should be $V_{N^2} = V_{N^1} \times (p_1 - p_2)$, from

Algorithm 5 Cuboid Score Function

Require: a point cloud $\mathcal{P}$; a shape model x.
Ensure: score σ.

1: $\sigma = 0$, $\mathcal{F}_{\text{bad}} =$ False, $\mathcal{I}_{\text{in}} = \varnothing$
2: **for** i =1 to 6 **do**
3: $\quad \mathcal{P}_{proj} = \text{projectPC}(\mathcal{P}; x, i)$
4: $\quad (\mathcal{F}_{\text{bad}}, \mathcal{I}^*_{\text{in}}) = \text{checkBound}(\mathcal{P}; \mathcal{P}_{proj}, x, i)$
5: $\quad$ **if** $\mathcal{F}_{\text{bad}}$ == True **then**
6: $\quad\quad$ **return**
7: $\quad$ **end if**
8: $\quad \mathcal{I}_{\text{in}} = \mathcal{I}_{\text{in}} \cup \mathcal{I}^*_{\text{in}}$
9: **end for**
10: $\sigma = |\mathcal{I}_{\text{in}}| \,/\, |\mathcal{P}|$

which a second plane $\mathcal{PL}^2$ is established. As done for the first plane, this inlier ratio r^2 gets computed (Step 12). If $\mathcal{PL}^1$ is the dominant or co-dominant plane, then the ratio r^1 should be greater than or close to r^2. Acceptable ratios are established based on the iteration count i and the historical record of ratios $\mathcal{B}$ from earlier iterations (Step 13). Should this relationship fail to hold, then the cuboid face generation process is cancelled. If it holds, then check that the other 7 points $p \in (P \setminus (p_1, p_2))$ lie to one side of $\mathcal{PL}^2$ as needed for the plane to define a face (Step 16). If $\mathcal{PL}^2$ separates points, then it is not an expected face and processing should move on to the next pair candidate.

If $\mathcal{PL}^1$ and $\mathcal{PL}^2$ pass the constraint checks, then the remaining planes $\mathcal{PL}^{3-6}$ are generated based on the normal vectors of $\mathcal{PL}^1$ and $\mathcal{PL}^2$, as well as the sampled points P (Step 17). $\mathcal{PL}^3$ is determined from V_{N^1} and the farthest point $p \in P$ to $\mathcal{PL}^1$, being the opposing face to $\mathcal{PL}^1$ and parallel to it. Similarly, $\mathcal{PL}^4$ is determined from V_{N^2} and the farthest point $p \in P$ to $\mathcal{PL}^2$, again being the opposing face to $\mathcal{PL}^2$ and parallel to it. The final two faces are orthogonal to both $\mathcal{PL}^1$ and $\mathcal{PL}^2$, thus $V_{N^5} = V_{N^1} \times V_{N^2}$. Recovery of the actual planes requires establishing points on each of them. Project the sampled points $p \in P$ onto $\mathcal{PL}^2$ to give p_{proj}, then compute the distance to the virtual plane going through the origin with the normal V_{N^5}. The points with extremal distances each lie on one of the final two faces. A point that maximizes the distance, and one that minimizes the distance, establish $\mathcal{PL}^5$ and $\mathcal{PL}^6$, respectively.

These 6 planes $\mathcal{PL}^{1-6}$ define the cuboid. The actual cuboid shape model parametrization x is given by the 8 corner points of the cuboid, which are intersection points of the planes $\mathcal{PL}^{1-6}$ (Step 18). The model x will be saved to the model set $\mathcal{X}$ (Step 19). The ratios r^1 and r^2 will be stored in $\mathcal{B}$ for future use (Step 20).

Score function. Algorithm 5 describes cuboid candidate model scoring, which is based on the distance from the planar surface patches defining the cuboid faces. For each face of the cuboid model x, project the point cloud points $p \in \mathcal{P}$ to the corresponding plane to get the projected points $\mathcal{P}_{proj}$ (Step 3). For those points $p \in \mathcal{P}$ which are near the plane, calculate the distances of their projections $p_{proj} \in \mathcal{P}_{proj}$ to the four edges of the face. If a large number of $p_{proj} \in \mathcal{P}_{proj}$ are far away from any edge or out of the face, the bad fitting flag $\mathcal{F}_{\text{bad}}$ will be set to True. If $\mathcal{F}_{\text{bad}}$ is marked as True, then the shape model x will be discarded (Step 5). Otherwise, the points will be added to a temporary inlier set $\mathcal{I}^*_{\text{in}}$ (Step 4). The final inlier set $\mathcal{I}_{\text{in}}$ will contain the points which can pass the bound check for any of the 6 faces (Step 8). The single score calculated is the ratio between the number of inliers $|\mathcal{I}_{\text{in}}|$ to the number of points in the point cloud $|\mathcal{P}|$ (Step 10).

REFERENCES

[1] L. Spreeuwers, "Fast and accurate 3D face recognition," *International journal of computer vision*, vol. 93, no. 3, pp. 389–414, 2011.

[2] P. H. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer vision and image understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[3] U. Mehmood, "Cuboid fit (ransac)," https://github.com/usamamehmood3/Cuboid-Fitting-RanSAC, 2021.