

Real-Time 3D Semantic Scene Perception for Egocentric Robots with Binocular Vision

Khang Nguyen

Tuan Dang

Manfred Huber

Abstract—Perceiving a three-dimensional (3D) scene with multiple objects while moving indoors is essential for vision-based mobile cobots, especially for enhancing their manipulation tasks. In this work, we present an end-to-end pipeline with instance segmentation, feature matching, and point-set registration for egocentric robots with binocular vision and demonstrate the robot’s grasping capability through the proposed pipeline. First, we design an RGB image-based segmentation approach for single-view 3D semantic scene segmentation, leveraging common object classes in 2D datasets to encapsulate 3D points into point clouds of object instances through corresponding depth maps. Next, 3D correspondences of two consecutive segmented point clouds are extracted based on matched keypoints between objects of interest in RGB images from the prior step. In addition, to be aware of spatial changes in 3D feature distribution, we also weigh each 3D point pair based on the estimated distribution using kernel density estimation (KDE), which subsequently gives robustness with less central correspondences while solving for rigid transformations between point clouds. Finally, we test our proposed pipeline on the 7-DOF dual-arm Baxter robot with a mounted Intel RealSense D435i RGB-D camera. The result shows that our robot can segment objects of interest, register multiple views while moving, and grasp the target object. The source code is available at https://github.com/mkhangg/semantic_scene_perception.

I. INTRODUCTION

Egocentric vision is crucial for machine and human vision, especially in a dense environment, due to its high selective attention to objects of interest while ignoring non-relevant entities (e.g., walls, floor, etc.) [1], [2]. From the standpoint of 3D perception for autonomous robots, the spatial information of objects of interest is also needed to ameliorate their manipulation tasks. Indeed, currently segmentation and registration tasks are often done separately from the *point cloud perspective* while considering every 3D point presented in the scene. Yet, deploying both procedures simultaneously may result in expensive computations on resource-constraint robots. For this reason, enabling a lightweight egocentric 3D segmentation, feature matching, and scene reconstruction pipeline is essential for vision-based indoor mobile cobots.

Considerable prior efforts are spent in learning matching features between images with deterministic algorithms [3]–[6] and with machine learning (ML) models [7], [8], which subsequently lead to robotics applications [9]–[12]. These works made notable contributions to 3D scene perception; however, 3D semantic scene perception for indoor mobile cobots also needs spatial occupancy information of objects

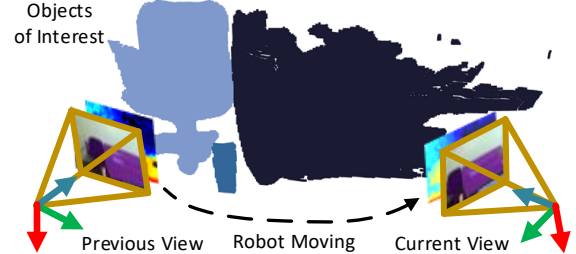


Fig. 1: Overview of 3D semantic scene perception for vision-based indoor robots with binocular vision and multiple egocentric views.

of interest, particularly in household settings, to initiate manipulation (e.g., appropriately grasping target objects).

Recently proposed methods mainly extract keypoints from images without focusing on objects of interest [13], [14], leading to expensive computation in later stages. Likewise, classical keypoint matching methods usually operate on entire images, but not regions of interest, which might result in mismatched or unwanted correspondences. Furthermore, the Iterative Closest Point-based (ICP) registration and ML-based methods for 3D multiview perform on a large set of 3D points without prior knowledge about the scene.

Therefore, to fill gaps in previous works and improve 3D semantic scene perception for vision-based mobile cobots as illustrated in Fig. 1, we (1) use lightweight semantic segmentation, which focuses only on objects of interest, (2) map 2D correspondences between two RGB images into 3D correspondences between two point clouds and align them with the awareness of important points, and (3) design a complete pipeline for indoor mobile robots with binocular vision to achieve 3D semantic perception (Fig. 2) by using an RGB-D camera in real-time along with robot operations.

As performing these tasks on a robot without a dedicated Graphics Processing Unit (GPU) requires computational power and memory optimization, we face a few challenges when designing this pipeline. Indeed, performing deep learning-based (DL) for semantic segmentation and matching point sets requires a huge consideration and customization of the state-of-the-art models, which are also included in the pipeline. Also, reconstructing a scene from multiple views is a complex iterative task since the traditional registration methods usually perform on the entire point cloud inputs.

In this work, we address the aforementioned challenges and make the following contributions: (1) a robust method to extract and statistically weigh 3D correspondences for rigid point cloud alignment, (2) an end-to-end segmentation, feature matching, and global registration pipeline for egocentric robots with binocular vision, (3) tests with a real robot system to verify the correctness of our proposed method.

All authors are with the Learning and Adaptive Robotics Laboratory, Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76013, USA. (emails: khang.nguyen8@mavs.uta.edu, tuan.dang@uta.edu, huber@cse.uta.edu)

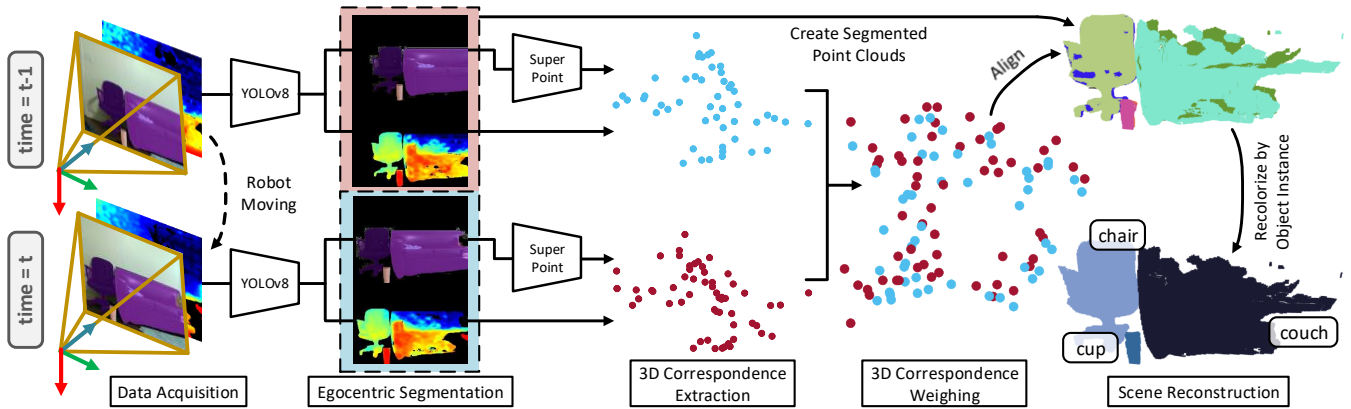


Fig. 2: The 3D semantic scene perception pipeline when the robot takes two (or multiple) views of a scene includes (1) egocentric segmentation to create point clouds of objects of interest (Sec. III), (2) extracting and matching corresponding features on masked RGB images to infer 3D correspondences via depth maps (Sec. IV), (3) finding optimal transformations based on weighted 3D correspondences and reconstructing the 3D scene (Sec. V), and (4) returning the aligned point cloud from multiple views with segmented objects.

II. RELATED WORK

Egocentric 3D Object Segmentation: Egocentric object segmentation has evolved rapidly in the past decade, from first-person view [15]–[17] to robotics applications [2], [18], [19]. Recently, this line of research has been enhanced into the domain of 3D computer vision, with DL models training on large computer-aided design (CAD) datasets [2], [18]. These works generally preclude themselves from segmenting objects that are unavailable in 3D datasets but present in 2D datasets. However, point cloud representations of objects, as well as scenes of multiple objects, in an egocentric view are composed of RGB and depth images from RGB-D cameras. Understanding these subcomponents and how point clouds are created, we leverage state-of-the-art segmentation models [20] to segment objects represented in RGB images and thus encapsulate 3D points into point clouds of the recognized objects when fusing depth images during the point cloud acquisition process. Therefore, this design takes advantage of the richness of object classes in the MS COCO image dataset but is unavailable in CAD datasets to recognize objects from an egocentric view.

Feature Detection & Matching: Traditional feature detection and extraction techniques, such as SIFT [3], SURF [4], and BRIEF [5], coupled with RANSAC [21] as an outlier rejection technique have been widely adopted in vision-based applications. Within this realm, ORB [10] is the first traditional feature-matching method to operate in real-time. Recent development in DL also introduces networks that serve feature-matching purposes, including TILDE [22], DeepDesc [23], LIFT [24], UCN [25], and SuperPoint [7] with its variants, such as SuperGlue [8] and LightGlue [26].

One closely similar work [14] proposes a SuperPoint-like framework with DepthNet for depth inference from RGB images to detect 3D keypoints for ego-motion estimation. However, the mentioned work does not maintain the real-time performance of SuperPoint due to the expensive computation of DepthNet. Additionally, 3D keypoint constructed matches from SuperPoint and depth maps generated by DepthNet lead to accumulated errors when encountering out-of-distribution scenes. To alleviate such expensive computa-

tions and potential errors, we form a 2D position embedding while training SuperPoint, overlay RGB images with segmentation masks to reduce the search space of the matching procedure, take corresponding depth frames without the need for DepthNet, and obtain 3D correspondences by mapping 2D correspondences produced by the retrained SuperPoint to depth maps that are also from the RGB-D image stream.

Point Cloud Alignment & Registration: Point cloud alignment and registration have predominantly depended on variants of the ICP algorithm, including point-to-point [27], [28] and point-to-plane [29] strategies. This area of study can be divided into global registration [30], [31], which primarily operates on 3D correspondence candidates based on point-to-point matches between point clouds, to find the optimal alignment, and local registration [27]–[29], [32], which requires an initial rough alignment to produce a refined alignment that tightly aligns two point clouds. However, global registration often requires multiple iterations to extract good correspondences. Likewise, local registration might lead to inaccurate results or does not converge to an optimal alignment if a rough alignment is not initialized. Therefore, to address the extensive search step for correspondence finding in global registration, we provide the point-set registration procedure with alternative 3D correspondences composed of matched features on RGB images and depth values on corresponding depth maps.

Robust kernels [33], [34] are proposed as an outlier rejection method for 3D correspondences and applied to point cloud alignment problems. Still, these techniques are highly sensitive to hyperparameter choices. To address the sensitivity induced by hyperparameters as in previous works, we utilize kernel density estimation (KDE) – a non-parametric statistic method to estimate the importance weight of each correspondence, which has recently been used for estimating each point’s likelihood [35], and as backbones of self-attention modules for density-aware 3D object detection in self-driving cars problems [36], [37]. Notably, KDE was also mathematically and experimentally proven to be a noise-free and robust registration method for single-object models based on their 3D correspondences [38] with offline experiments.

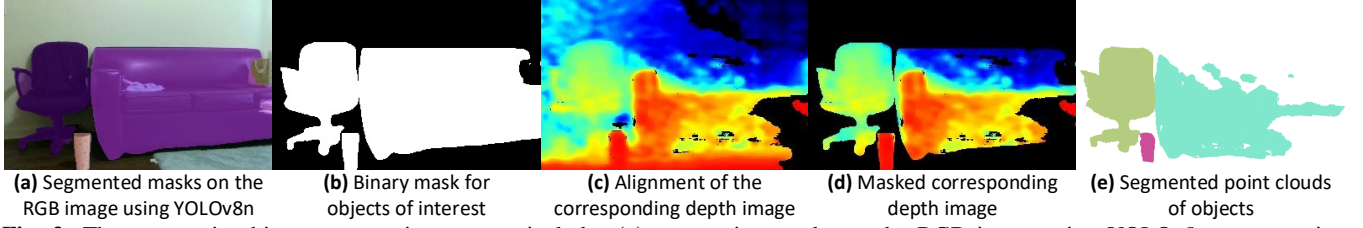


Fig. 3: The egocentric object segmentation process includes (a) segmenting masks on the RGB image using YOLOv8n segmentation model, (b) obtaining and aggregating binary masks of the objects of interest, (c) aligning the corresponding depth image, (d) rectifying non-masked depth pixels on the aligned depth image with obtained masks, and (e) creating point clouds of such objects.

Algorithm 1: Egocentric Object Segmentation

Input : $\mathcal{D} :=$ raw depth image, $\mathcal{I}(w, h) :=$ raw RGB image
 $I^D, I^{RGB} :=$ depth & RGB cameras intrinsic matrices
 $R, T :=$ extrinsic matrix btw RGB & depth cameras

Output: $\mathcal{P}_M = \{\mathbf{p}_i\} = \{x_i, y_i, z_i\} :=$ segmented point cloud

```

1 function ego_segment( $\mathcal{D}, \mathcal{I}, I^D, I^{RGB}, R, T$ )
2    $\mathcal{D}_M, \mathcal{P}_M = [], []$ 
3    $\mathcal{M}_I = \text{segment}(\mathcal{I})$ 
4    $\mathcal{D}_{\text{filled}} = \text{fill\_holes}(\mathcal{D}, \text{nearest\_from\_around})$ 
5    $\mathcal{D}_{\text{align}} = \text{align\_frame}(\mathcal{D}_{\text{filled}}, \mathcal{I}, I^D, I^{RGB}, R, T)$ 
6   for  $u \in w$  and  $v \in h$  do
7      $\mathcal{D}_M(u, v) = \mathcal{D}_{\text{align}}(u, v)$  if  $\mathcal{M}_I(u, v) \neq 0$  else 0
8   for  $m \in w$  and  $n \in h$  do
9     if  $\mathcal{D}_M(m, n) \neq 0$  then
10       $z_i = \mathcal{D}_{\text{rect}}(m, n)$ 
11       $x_i, y_i = (m - c_x^D) \cdot z_i / f_x^D, (n - c_y^D) \cdot z_i / f_y^D$ 
12       $\mathcal{P}_M.append(\text{create\_point}(x_i, y_i, z_i))$ 
13    $\mathcal{P}_M = \text{remove\_outliers}(\mathcal{P}_M)$ 
14   return  $\mathcal{P}_M$ 

```

III. EGOCENTRIC 3D OBJECT SEGMENTATION

In this section, we propose an algorithm to egocentrically segment objects in RGB-D frames, as described in Alg. 1 with w and h respectively denoted as width and height of RGB images, typically $w = 320$ and $h = 240$. We denote the depth image taken from depth cameras as \mathcal{D} , the RGB image taken from the RGB camera as \mathcal{I} , and a point cloud with n points as $\mathcal{P} := \{\mathbf{p}_i\}$, for $i = 1, 2, \dots, n$, where $\mathbf{p}_i = [x_i, y_i, z_i]^T$ represent points of \mathcal{P} in Cartesian coordinates.

As shown in Fig. 3 and Alg. 1, the segmentation process first takes \mathcal{D} and \mathcal{I} from the image stream, segments objects of interest presented in \mathcal{I} (Fig. 3a) to obtain object's masks \mathcal{M}_I (Fig. 3b). Next, to guarantee the resulted \mathcal{P}_M 's quality, \mathcal{D} is holes-filled using the value from the neighboring pixel closest to the camera, which results in $\mathcal{D}_{\text{filled}}$. Hence, $\mathcal{D}_{\text{filled}}$ is dimensionally aligned with respect to \mathcal{I} (Fig. 3c) with the preprogrammed RGB-D camera's intrinsic and extrinsic parameters, I^D, I^{RGB}, R , and T . The aligned depth frame, $\mathcal{D}_{\text{align}}$, is then processed pixel-wise to rectify depth pixels that are outside of \mathcal{M}_I (Fig. 3d) and \mathcal{D}_M is pixel-wise converted into points in \mathcal{P}_M (Fig. 3e) based on optical centers, (c_x^D, c_y^D) , and focal lengths, (f_x^D, f_y^D) , of depth cameras. Lastly, \mathcal{P}_M is cleaned by removing possible outliers that resulted from holes in depth images in the masks' precision in the earlier step. Note that this procedure also requires skipping a few initial frames (about 5 frames) for the camera to guarantee auto-exposure with brightness control. The computational cost of this procedure on the real robot system is evaluated in Sec. VI-C.

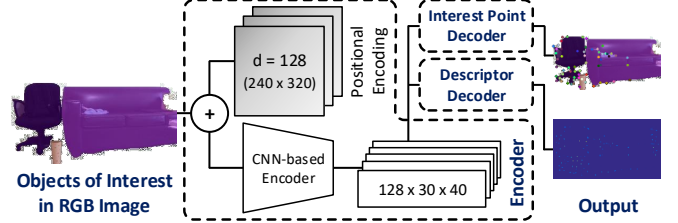


Fig. 4: The architecture of the feature-extraction network with 2D-positional embedding, where \oplus denotes embedding notation.

IV. FEATURE DETECTION & MATCHING

A. Network Architecture

Due to the positions of pixels as well as features varying among images in training datasets and in real-world scenes, we apply 1D-positional embedding into the 2D domain to improve the feature-extraction learning process with embedded positional information of every pixel presented on the input RGB image. The formula for a 2D-positional embedding encoder is given below:

$$\begin{cases} \psi(x, y, 2i) = \sin\left(\frac{x}{D}\right), & \psi\left(x, y, 2j + \frac{d}{2}\right) = \sin\left(\frac{y}{D}\right) \\ \psi(x, y, 2i + 1) = \cos\left(\frac{x}{D}\right), & \psi\left(x, y, 2j + 1 + \frac{d}{2}\right) = \cos\left(\frac{y}{D}\right) \end{cases} \quad (1)$$

where ψ represents the embedder, (x, y) are pixel coordinates, $D = 10,000^{1-(d/4)^{-1}}$ with d indicating the embedding space's dimension, and (i, j) are integers in $[0, d/4)$.

Applying Eq. 1 as shown in Fig. 4, we design our feature-extraction network with the shared 2D-positional encoder, feed-forwarding to interest point and descriptor decoders. For interest point and descriptor decoders, we inherit these modules from the vanilla SuperPoint [7]. The training details and the qualitative results of this network design on the HPatches dataset [39] are in Sec. VI-A.

B. Feature Detection & Matching Strategy

Since the vanilla approach of SuperPoint takes in entire RGB frames as inputs, it might mismatch a feature on one object in the previous frame to a closely similar feature on another object in the current frame. To eliminate the effect of this, we also leverage the segmented mask in Sec. III to provide SuperPoint with masked RGB images of corresponding objects between RGB frames (Fig. 4), guaranteeing feature-scanning regions within masked regions.

As shown in Fig. 5, we first create masked RGB images of each corresponding object in two consecutive frames and then apply the retrained SuperPoint on each pair of images (e.g., cups, couches, and chairs as illustrated) to extract and

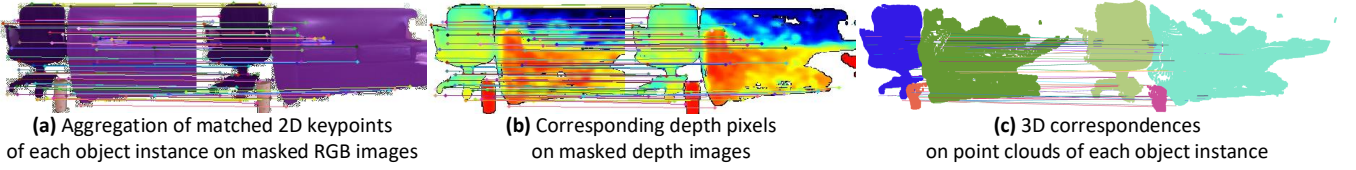


Fig. 5: The 3D correspondences matching process includes (a) extracting and matching keypoints between masked RGB images, (b) finding corresponding depth pixels on rectified depth images, and (c) mapping 3D correspondences between point clouds of object instances.

match 2D keypoints within each object instance. This avoids feature mismatching across non-corresponding objects. We then aggregate matched features from separated objects in the previous step (Fig. 5a), and find the corresponding matched pixels in the aligned depth images (Fig. 5b). With the image coordinates in Fig. 5a and depth values from Fig. 5b, we compute 3D correspondences between point clouds using line 11 in Alg. 1. The correspondences between point clouds of each object instance presented in the scene are shown in Fig. 5c.

V. POINT CLOUD ALIGNMENT & REGISTRATION

In this section, we elaborate on the point cloud alignment procedure with extracted 3D correspondences between \mathcal{P}_t and \mathcal{P}_{t-1} from Sec. IV. Without loss of generality, we denote 3D corresponding point sets in the current view and previous view as \mathcal{K}_t and \mathcal{K}_{t-1} , respectively, and each 3D pointset of m points $\mathcal{K}_i := \{\mathbf{p}_j^{\mathcal{K}_i}\}$, for $j = 1, 2, \dots, m$, where $\mathbf{p}^{\mathcal{K}_i}$ are points of \mathcal{K} in Cartesian coordinates, with $m \ll n$.

A. 3D Correspondence Importance Weighting

Noticing that the feature detection process (Sec. IV) mainly concentrates on segmented regions on RGB images from Sec. III, these procedures produce \mathcal{K}_t (or \mathcal{K}_{t-1}) that are bounded in several regions in 3D space, which conceptually is a mixture of distributions with dense regions, which are dense with points, and outliers with respect to the spatial distribution of \mathcal{K}_t .

1) *Weight Initialization:* To weigh each correspondence in \mathcal{K}_t and \mathcal{K}_{t-1} , we first initialize that point's weight via its number of neighboring points inside a specific radius and then compute the cumulative density of points within a region in a dimension-wise sense to obtain the weight of that point.

2) *Density Estimation:* The density of the unknown distribution of \mathcal{K}_t is estimated using (i) KDE and (ii) Improved Sheather-Jones (ISJ) algorithm for data-driven optimal bandwidth selection [40] since we (i) could not make any assumptions toward the distribution of \mathcal{K}_t and (ii) would like our transformation in Sec. V-B to be robust with outliers, which are less-relevant points in \mathcal{K}_t .

This procedure is illustrated in Alg. 2. We first initialize the weighing coefficient, w_i , for each $\mathbf{p}_i^{\mathcal{K}_t}$, as the number of neighboring points, $|\mathcal{N}_i|$, within a radius, r . Next, to avoid the curse of dimensionality when applying non-parametric methods and since x , y , and z of each $\mathbf{p}_i^{\mathcal{K}_t}$ are independent of each other in our problem, we treat them as three random variable vectors X , Y , and Z . Taking X as an illustration, we apply KDE for each random-variable vector as follows:

$$\hat{f}_X(x) = \frac{1}{mH} \sum_{i=1}^m w_i \cdot K \left[\frac{x - x_i}{H} \right] \quad (2)$$

Algorithm 2: Important Points Weighting

Input : $\mathcal{K}_t :=$ set of 3D corresponding points of \mathcal{P}_t
 $r :=$ radius of sphere around a 3D point
Output: $W = \text{diag}(w_1, \dots, w_n) :=$ diagonal weighing matrix

1 **function** weigh_corresponding_points(\mathcal{K}_t, r)
2 $w, F = [], []$
3 $w.append(|\mathcal{N}_i| = \text{neighbor}(\mathbf{p}_i^{\mathcal{K}_t}, r))$ **for** $\mathbf{p}_i^{\mathcal{K}_t} \in \mathcal{K}_t$
4 **for** $k \in \{\mathcal{K}_t[:, 0], \mathcal{K}_t[:, 1], \mathcal{K}_t[:, 2]\}$ **do**
5 $F.append(\text{fft.KDE}(k, w, H = \text{optband.ISJ}(k)))$
6 $w = w \cdot f_i$ **for** $f_i \in F$
7 **return** $\text{diag}(w^T)$

where H represents the optimal bandwidth for the kernel, K , which is often chosen as the Gaussian distribution.

To reduce computational complexity of $\mathcal{O}(m^2)$ from Eq. 2, the binning (alternative) approximation for Eq. 2 is introduced [40]. Written in terms of grid points $\{\mathbf{g}_j\} \in [\mathbf{g}_1, \mathbf{g}_M]$ covering all $\mathbf{p}_i^{\mathcal{K}_t}$, and grid counts $\{c_j\}$ designating weights chosen to represent the number of $\mathbf{p}_i^{\mathcal{K}_t}$'s that are near \mathbf{g}_j , for $j = 1, 2, \dots, M$, and $M \neq m$, yields:

$$\widetilde{f_{X, \mathbf{g}_j}} := \tilde{f}_X(\mathbf{g}_j) = \frac{1}{mH} \sum_{i=1}^M w_i c_i \cdot K \left[\frac{\mathbf{g}_j - \mathbf{g}_i}{H} \right] \approx \hat{f}_X(x) \quad (3)$$

Taking advantage of the FFT's time complexity of $\mathcal{O}(M \log M)$, Eq. 3 is then rewritten in the form of convolution as:

$$\widetilde{f_{X, \mathbf{g}_j}} = \sum_{i=-(M-1)}^{M-1} c_{j-i} \mathbf{k}_i \text{ with } \mathbf{k}_i = \frac{w_i}{m} \cdot K \left[\frac{(\mathbf{g}_M - \mathbf{g}_1)i}{H} \right] \quad (4)$$

Together with the binning process in Eq. 3, the time complexity of Alg. 2 is $\mathcal{O}(m + M \log M)$. The optimal bandwidth, H , is obtained from the ISJ algorithm with \mathcal{K}_t .

3) *Weight Updating:* Applying Eq. 4, we obtain density functions \hat{f}_X , \hat{f}_Y , and \hat{f}_Z for X , Y , and Z , respectively, as illustrated in Fig. 6a. Hence, w_i for each $\mathbf{p}_i^{\mathcal{K}_t} := \{x_i, y_i, z_i\}$ is updated from the naive estimation in Sec. V-A.1:

$$w_i := w_i \cdot \hat{f}_X(x_i) \cdot \hat{f}_Y(y_i) \cdot \hat{f}_Z(z_i) \quad (5)$$

Using Eq. 5 for all points in \mathcal{K}_t , we obtain the weighing $m \times m$ diagonal matrix, $W = \text{diag}(w_1, \dots, w_m)$. The weights are illustrated in Fig. 6b, with bigger points portraying larger weights, and vice versa.

B. Rigid Motion for Point Cloud Alignment

To find a rigid transformation matrix, T , for \mathcal{P}_t and \mathcal{P}_{t-1} , that aligns \mathcal{K}_t and \mathcal{K}_{t-1} in \mathbb{R}^3 , we define the optimizer in terms of a rotation matrix, R , a translation vector, \mathbf{t} , and the weighing coefficient, w_i for $\mathbf{p}_i^{\mathcal{K}_t}$ and $\mathbf{p}_i^{\mathcal{K}_{t-1}}$, as follows:

$$T = [R, \mathbf{t}] = \underset{R, \mathbf{t}}{\text{argmin}} \sum_{i=1}^n w_i \left\| \mathbf{p}_i^{\mathcal{K}_t} - (R \mathbf{p}_i^{\mathcal{K}_{t-1}} + \mathbf{t}) \right\|^2 \quad (6)$$

From Eq. 6, the optimal 3×1 translation vector, \mathbf{t} , is:

$$\mathbf{t} = \overline{\mathbf{p}^{\mathcal{K}_t}} - R \cdot \overline{\mathbf{p}^{\mathcal{K}_{t-1}}} \quad (7)$$

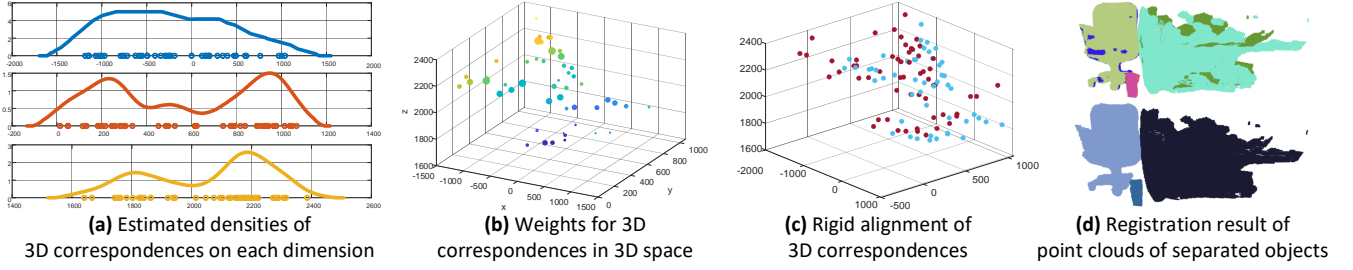


Fig. 6: The point cloud alignment process includes (a) dimension-wise estimating densities of 3D correspondences along x-axis (blue), y-axis (orange), and z-axis (yellow), (b) computing weights for 3D correspondences, (c) solving for the optimal rigid transformation based on 3D correspondences and their weights, and (d) aligning point clouds (top) with re-colorization based on each object instance (down).

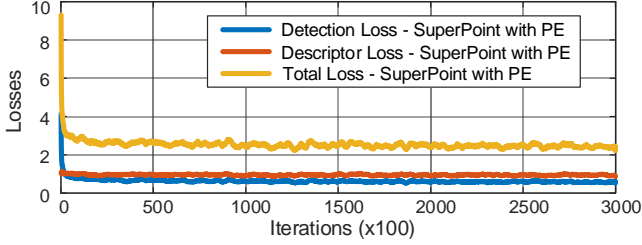


Fig. 7: Training performance of SuperPoint with positional embedding, including detection loss, descriptor loss, and total loss.



Fig. 8: Qualitative results of Superpoint with positional embedding with brightness and viewpoint change images from HPatches.

$$\text{where } \bar{\mathbf{p}}^{\mathcal{K}_t} := \frac{\sum_{i=1}^n w_i \mathbf{p}_i^{\mathcal{K}_t}}{\sum_{i=1}^n w_i} \text{ and } \bar{\mathbf{p}}^{\mathcal{K}_{t-1}} := \frac{\sum_{i=1}^n w_i \mathbf{p}_i^{\mathcal{K}_{t-1}}}{\sum_{i=1}^n w_i}$$

are weighted centroids of \mathcal{K}_t and \mathcal{K}_{t-1} , respectively.

Meanwhile, to find the optimal 3×3 rotation matrix, R , we first compute the covariance matrix, C , which is the product of the $3 \times m$ matrix, D_t , column-wise containing deviations of $\mathbf{p}_i^{\mathcal{K}_t}$ from $\bar{\mathbf{p}}^{\mathcal{K}_t}$ in \mathcal{K}_t and the $3 \times m$ matrix, D_{t-1} , column-wise containing deviations of $\mathbf{p}_i^{\mathcal{K}_{t-1}}$ from $\bar{\mathbf{p}}^{\mathcal{K}_{t-1}}$ in \mathcal{K}_{t-1} , and W obtained from Alg. 2 in Sec. V-A:

$$C = D_t W D_{t-1}^T \quad (8)$$

Hence, the optimal rotation matrix is computed in terms of rotation matrices, U and V , resulting from the singular value decomposition $U \Sigma V^T$ of C from Eq. 8, as follows:

$$R = V \begin{bmatrix} 1 & & \\ & \ddots & \\ & & \det(VU^T) \end{bmatrix} U^T \quad (9)$$

Note that the scaling matrix, Σ , is omitted in Eq. 9 as we need to preserve object sizes and shapes in \mathcal{P}_{t-1} in Eq. 11.

Therefore, from Eq. 7 and Eq. 9, the 4×4 homogeneous rigid transformation matrix solving Eq. 6 for \mathcal{K}_t and \mathcal{K}_{t-1} and eventually for \mathcal{P}_t and \mathcal{P}_{t-1} , is defined as:

$$T = \begin{bmatrix} R & \mathbf{t} \\ 0_{1,3} & 1 \end{bmatrix} \quad (10)$$

With the homogeneous transformation, T , from Eq. 10, we align two multiview point clouds, \mathcal{P}_t and \mathcal{P}_{t-1} , as follows:

$$\mathcal{P}_{\text{align}} = \mathcal{P}_t + \left[R \mathbf{p}_i^{\mathcal{P}_{t-1}} + \mathbf{t} \right]_{i=1}^n := \mathcal{P}_t + T(\mathcal{P}_{t-1}) \quad (11)$$

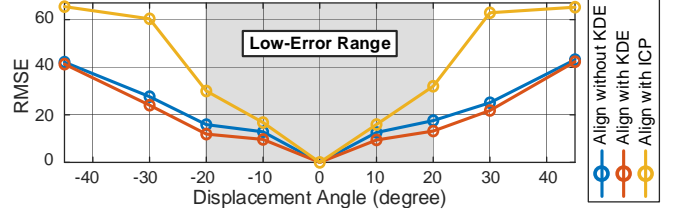


Fig. 9: Alignment error computed in RMSE at multiple angles of alignments with/without using KDE, and point-to-point ICP.

VI. EVALUATION & EXPERIMENTAL RESULTS

A. Performance of SuperPoint with Positional Embedding

We retrain SuperPoint with 2D positional embedding with $d = 128$ (Sec. IV) on the MS COCO 2014 dataset with interest points labeled by MagicPoint, which is refined on a synthetic auto-generated dataset (Fig. 7). All images are resized to 240×320 and are augmented by adding random brightness and contrast, Gaussian noises, shades, and motion blur. The training process takes place on the NVIDIA RTX 4090 GPU with 10 epochs (300,000 iterations).

Also, Fig. 8 shows qualitative results of SuperPoint with positional embedding on images from HPatches [39], illustrating the robustness in the two most common scenarios: brightness change (Fig. 8a) and viewpoint change (Fig. 8b).

B. Point Cloud Alignment Error at Multiple Angles

We evaluate the performance of our point cloud alignment (Sec. V-B) by calculating Root Mean Squared Error (RMSE) between the two correspondence sets, \mathcal{K}_{t-1} and \mathcal{K}_t . To do this, we move the camera to various angles with a 2-meter distance from the scene on a planar surface, including 0° (stands at its initial position), $\pm 10^\circ$, $\pm 20^\circ$, $\pm 30^\circ$, and $\pm 45^\circ$, and computing their alignment using RMSE.

As shown in Fig. 9, RMSE is larger as the displacement angle gets wider. The results also prove that KDE is effectively involved in the process of reducing alignment errors. Meanwhile, ICP registration underperforms our alignment method, as the rough alignment assumption is not guaranteed when capturing views at a wide displacement angle.

C. Deployment on Baxter Robot

1) *Experimental Setup:* To obtain RGB-D images at multiple views, we mount the Intel RealSense D435i RGB-D camera on the Baxter robot with a setup scene of a table, a chair, a bag, and two plastic cups on the table. Prior to the experiment, the robot is pre-calibrated using the multiplanar self-calibration method [41] for precise object manipulation.

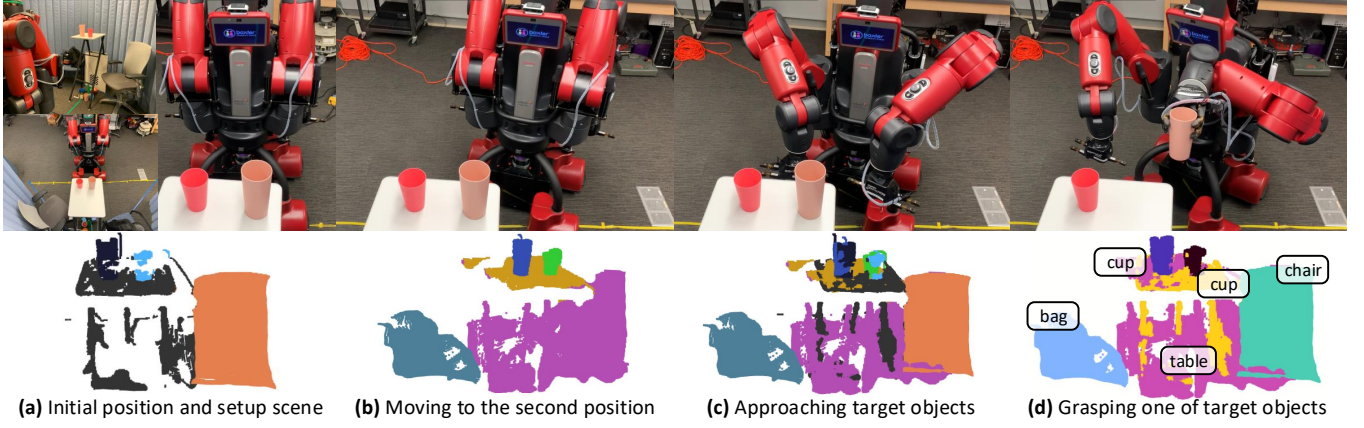


Fig. 10: Experiment setup with (top) the Baxter (a) observing the scene at its first view, (b) moving to and capturing the second scene of 20-degree displacement, (c) approaching the target objects, and (d) grasping one of the cups; and (down) each-step results in the pipeline.

2) Robot Moving & Capturing Scene at Multiple Views:

To take multiple views of the scene, the Baxter first stands in one position, takes one view (Fig. 10a), moves to another angle, and takes another view (Fig. 10b), with the robot’s movements supported by the Dataspeed mobility base.

Specifically, the process takes place on two separate computers, including one acquiring data from the RGB-D camera and the main computer, with synchronization using the Robot Operating System (ROS) messages through TCP/IP protocols between them. At each position, once the computer in charge of the robot’s vision receives a motion command from the main computer, it executes motion planning and notifies the main computer when the motion command is done. Then the main computer captures a point cloud from the RGB-D camera. The process is then repeated for other angles, which provides the knowledge of the scene from multiple views.

3) *Segmenting & Aligning Multiview Point Clouds:* After capturing multiview point clouds, the Baxter first egocentrically segment objects in the scene (Sec. III), match 3D correspondences between two views (Sec. IV), solves for the transformation of weighted 3D correspondences for rigid alignment (Sec. V), and eventually gives itself a scene understanding of the bag in the second view, the chair in the first view, and the table and the cups in both views.

4) *Approaching & Grasping Target Objects:* To demonstrate the feasibility of robotic grasping with 3D semantic scene perception for cobot-like systems, we navigate the Baxter to approach the target objects in the scene. When the target objects are within the robot’s workspace (Fig. 10c), the Baxter can grasp those objects efficiently (Fig. 10d).

| Task | Device | Time Complexity | Runtime |
|-------------------------|--------|-----------------------------|-----------|
| Egocentric Segmentation | GPU | n/a | 96.49 ms |
| Keypoint Extraction | CPU | n/a | 158.91 ms |
| Keypoint Matching | CPU | n/a | 2.35 ms |
| Keypoint Weighting | CPU | $\mathcal{O}(m + M \log M)$ | 2.07 ms |
| Point Cloud Alignment | CPU | $\mathcal{O}(m^3)$ | 0.73 ms |

TABLE I: Time complexity and runtime in milliseconds (ms) of segmentation, keypoint extraction and matching, keypoint weighting, and point cloud alignment, on onboard processing units.

5) *Time Complexity on Conventional Hardware:* We use the OpenVINO library [42] to deploy YOLOv8n on the native onboard GPU (Intel HD Graphics 4000) of the Intel

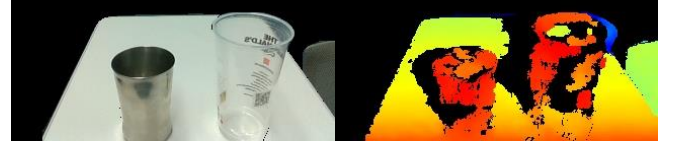


Fig. 11: Failure of depth perception of transparent and shiny cups. NUC5i7RYH PC – the computer processing RGB-D images – on our customized software framework [43]. The average runtime across onboard devices is reported in Table I.

D. Demonstration

The video demonstrates a scenario with the Baxter (1) standing in one position, (2) moving to a position of 20° displacement while (3) performing the pipeline, and (4) grasping the plastic cup: <https://youtu.be/-dho71r56U>.

VII. LIMITATIONS & FUTURE WORKS

As we specifically aim for indoor robots with RGB-D cameras, our primary limitation is its inapplicability to far-field omnidirectional perception (*e.g.*, LiDAR-based perception). In addition, the depth camera also fails to recognize transparent and shiny objects, as illustrated in Fig. 11, which makes the robot grasping inaccurate. Therefore, we reserve the tasks of perceiving transparent/shiny objects and aligning RGB-D cameras and LiDAR sensors for future work.

VIII. CONCLUSIONS

This work introduces an end-to-end pipeline with instance segmentation, feature matching, and alignment for mobile cobots with RGB-D perception. Our pipeline first segments objects of interest in the scene as the robot moves and matches features between consecutive RGB images before obtaining 3D correspondences via depth maps. The 3D correspondences are then statistically weighted based on their spatial distribution using KDE for rigid point cloud alignment. Through real-world experiments on the 7-DOF dual-arm Baxter robot with an Intel RealSense D435i RGB-D camera, the results show the robot is able to semantically understand the setup scene and grasp the target objects.

IX. ACKNOWLEDGMENT

We thank Nghia Le (University of Toronto) for fruitful discussions on selecting non-parametric statistical methods.

REFERENCES

- [1] L. B. Smith, S. Jayaraman, E. Clerkin, and C. Yu, “The developing infant creates a curriculum for statistical learning,” *Trends in cognitive sciences*, vol. 22, no. 4, pp. 325–336, 2018.
- [2] G. Humblot-Renau, L. Marchegiani, T. B. Moeslund, and R. Gade, “Navigation-oriented scene understanding for robotic autonomy: learning to segment driveability in egocentric images,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2913–2920, 2022.
- [3] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, pp. 91–110, 2004.
- [4] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *Computer Vision—ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. Springer, 2006, pp. 404–417.
- [5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *Computer Vision—ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010. Proceedings, Part IV 11*. Springer, 2010, pp. 778–792.
- [6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.
- [7] D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superpoint: Self-supervised interest point detection and description,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.
- [8] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, “SuperGlue: Learning feature matching with graph neural networks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4938–4947.
- [9] C. Tomasi and T. Kanade, “Shape and motion from image streams under orthography: a factorization method,” *International journal of computer vision*, vol. 9, pp. 137–154, 1992.
- [10] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “Orb-slam: a versatile and accurate monocular slam system,” *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [11] S. Kelly, A. Riccardi, E. Marks, F. Magistri, T. Guadagnino, M. Chli, and C. Stachniss, “Target-aware implicit mapping for agricultural crop inspection,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9608–9614.
- [12] L. Lobefaro, M. V. Malladi, O. Vysotska, T. Guadagnino, and C. Stachniss, “Estimating 4d data associations towards spatial-temporal mapping of growing plants for agricultural robots,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 4212–4218.
- [13] Y.-Y. Jau, R. Zhu, H. Su, and M. Chandraker, “Deep keypoint-based camera pose estimation with geometric constraints,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4950–4957.
- [14] J. Tang, R. Ambrus, V. Guizilini, S. Pillai, H. Kim, P. Jensfelt, and A. Gaidon, “Self-supervised 3d keypoint learning for ego-motion estimation,” in *Conference on Robot Learning*. PMLR, 2021, pp. 2085–2103.
- [15] A. Fathi, X. Ren, and J. M. Rehg, “Learning to recognize objects in egocentric activities,” in *CVPR 2011*. IEEE, 2011, pp. 3281–3288.
- [16] Y. Li, Z. Ye, and J. M. Rehg, “Delving into egocentric actions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 287–295.
- [17] V. Tschernezki, D. Larlus, and A. Vedaldi, “Neuraldiff: Segmenting 3d objects that move in egocentric videos,” in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 910–919.
- [18] A. Kundu, Y. Li, and J. M. Rehg, “3d-rcnn: Instance-level 3d object reconstruction via render-and-compare,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3559–3568.
- [19] C. Li, Y. Meng, S. H. Chan, and Y.-T. Chen, “Learning 3d-aware egocentric spatial-temporal interaction via graph convolutional networks,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8418–8424.
- [20] G. Jocher, A. Chaurasia, and J. Qiu, “YOLO by Ultralytics,” Jan. 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [21] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [22] Y. Verdie, K. Yi, P. Fua, and V. Lepetit, “Tilde: A temporally invariant learned detector,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5279–5288.
- [23] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, “Discriminative learning of deep convolutional feature point descriptors,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 118–126.
- [24] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, “Lift: Learned invariant feature transform,” in *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016. Proceedings, Part VI 14*. Springer, 2016, pp. 467–483.
- [25] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker, “Universal correspondence network,” *Advances in neural information processing systems*, vol. 29, 2016.
- [26] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys, “Lightglue: Local feature matching at light speed,” *arXiv preprint arXiv:2306.13643*, 2023.
- [27] P. J. Besl and N. D. McKay, “Method for registration of 3-d shapes,” in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.
- [28] Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.
- [29] S. Rusinkiewicz and M. Levoy, “Efficient variants of the icp algorithm,” in *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 2001, pp. 145–152.
- [30] 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.
- [31] Q.-Y. Zhou, J. Park, and V. Koltun, “Fast global registration,” in *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016. Proceedings, Part II 14*. Springer, 2016, pp. 766–782.
- [32] J. Park, Q.-Y. Zhou, and V. Koltun, “Colored point cloud registration revisited,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 143–152.
- [33] P. Babin, P. Giguere, and F. Pomerleau, “Analysis of robust functions for registration algorithms,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1451–1457.
- [34] N. Chebrolu, T. Labe, O. Vysotska, J. Behley, and C. Stachniss, “Adaptive robust kernels for non-linear least squares problems,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2240–2247, 2021.
- [35] W. Wu, Z. Qi, and L. Fuxin, “Pointconv: Deep convolutional networks on 3d point clouds,” in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2019, pp. 9621–9630.
- [36] J. S. Hu, T. Kuai, and S. L. Waslander, “Point density-aware voxels for lidar 3d object detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8469–8478.
- [37] S. Wang, K. Lu, J. Xue, and Y. Zhao, “Da-net: Density-aware 3d object detection network for point clouds,” *IEEE Transactions on Multimedia*, 2023.
- [38] Y. Tsin and T. Kanade, “A correlation-based approach to robust point set registration,” in *Computer Vision—ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part III 8*. Springer, 2004, pp. 558–569.
- [39] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, “Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5173–5182.
- [40] M. Wand, “Fast computation of multivariate kernel estimators,” *Journal of Computational and Graphical Statistics*, vol. 3, no. 4, pp. 433–445, 1994.
- [41] T. Dang, K. Nguyen, and M. Huber, “Multiplanar self-calibration for mobile cobot 3d object manipulation using 2d detectors and depth estimation,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 1782–1788.
- [42] OpenVINO, “OpenVINO GitHub,” 2023. [Online]. Available: <https://github.com/openvinotoolkit/openvino>
- [43] T. Dang, K. Nguyen, and M. Huber, “Perfc: An efficient 2d and 3d perception software-hardware framework for mobile cobot,” in *The International FLAIRS Conference Proceedings*, vol. 36, 2023.