IUST
Iran University of Science and Technology

Iranian Journal of Electrical and Electronic Engineering

Journal Homepage: ijeee.iust.ac.ir

Grasp Area Detection for 3D Object using Enhanced Dynamic Graph Convolutional Neural Network

Haniye Merrikhi*, Hossein Ebrahimnezhad*(C.A.)

Abstract: Robots have become integral to modern society, taking over both complex and routine human tasks. Recent advancements in depth camera technology have propelled computer vision-based robotics into a prominent field of research. Many robotic tasks—such as picking up, carrying, and utilizing tools or objects—begin with an initial grasping step. Vision-based grasping requires the precise identification of grasp locations on objects, making the segmentation of objects into meaningful components a crucial stage in robotic grasping. In this paper, we present a system designed to detect the graspable parts of objects for a specific task. Recognizing that everyday household items are typically grasped at certain sections for carrying, we created a database of these objects and their corresponding graspable parts. Building on the success of the Dynamic Graph CNN (DGCNN) network in segmenting object components, we enhanced this network to detect the graspable areas of objects. The enhanced network was trained on the compiled database, and the visual results, along with the obtained Intersection over Union (IoU) metrics, demonstrate its success in detecting graspable regions. It achieved a grand mean IoU (gmIoU) of 92.57% across all classes, outperforming established networks such as PointNet++ in part segmentation for this dataset. Furthermore, statistical analysis using analysis of variance (ANOVA) and T-test validates the superiority of our method.

Keywords: Robotic Grasp, Grasp Area, Point Cloud, Part Segmentation, Dynamic Graph CNN

1 Introduction

ROBOTIC manipulation has emerged as a significant area of study, especially as robots increasingly replace humans in industries, services, and agriculture [1]–[6]. Grasping is central to many robotic tasks; however, robots are still markedly less adept than humans at grasping items [7], [8]. The advent of sophisticated depth and stereo vision cameras has fueled interest in vision-based grasping systems. Sahbani et al. [9] divided vision-based grasping methods into two categories: analytical and empirical (or data-driven) methods. Analytical methods, which were initially the dominant approach, rely on the analysis of geometry, kinematics, and dynamics to determine suitable grasps, such as force-closure, which ensures stability [9]–[12]. The primary advantage of these approaches is their ability to calculate and guarantee theoretical requirements for grasp quality. However, they have significant drawbacks, including the necessity for precise object information and high computational costs [13]. In contrast, data-driven strategies focus on categorizing various types of grasps and utilize learning techniques to enhance their effectiveness. Learning-based methods automatically configure grasp detection without human intervention, thereby reducing both programming efforts and computational costs [14]. Although data-driven approaches do not guarantee successful grasping and require large volumes of data, the collection of comprehensive databases and recent advances in deep learning have increased interest in these methods [10], [13], [15].

In task-oriented grasping, the configuration of the

Iranian Journal of Electrical & Electronic Engineering, 2024.
Paper first received 28 September 2024 and accepted 22 December 2024.
* The authors are with the Department of Electrical Engineering, Sahand University of Technology, Tabriz, Iran.
E-mails: ebrahimnezhad@sut.ac.ir
Corresponding Author: Hossein Ebrahimnezhad

grasp is determined by the intended action or purpose of the grasp. The task constraints associated with a specific grasp can be encapsulated in the concept of affordance.[16]. Indeed, a grasp affordance describes how an object should be grasped to accomplish a specific task [17]. Different grasp affordances imply that the selection of graspable locations on an object will vary. For instance, to open a bottle, you typically grasp the cap. Thus, various affordances can be linked to different parts of an object. Segmenting objects based on these affordances is a vital process in the field of robotic grasping. In this work, we focus on segmenting household objects to facilitate grasping for everyday use.

Point clouds are dispersed collections of points in two or three dimensions that provide a versatile geometric representation, essential for numerous computer graphics and vision applications. They come from 3D data gathering tools like LiDAR scanners and RGB-D cameras [8], [18]. With advancements in 3D scanning technologies, the rapid acquisition of 3D point clouds has become increasingly feasible, making these representations more accessible for various applications [1], [19]. Despite their simplicity, point clouds lack inherent topological information, presenting challenges for effective processing and analysis. Traditional methods have relied on handcrafted features for point cloud analysis; however, the success of deep neural networks in image processing has sparked interest in adapting these techniques for point clouds [19]. Nonetheless, applying deep neural networks, such as convolutional neural networks (CNNs), to point clouds presents difficulties due to the naturally irregular structure of point clouds, in contrast to images [20].

Part-based grasping has gained significant attention in recent years due to its two key advantages: reducing the search space for grasp parameters and facilitating task-oriented grasping [17]. Consequently, segmenting objects to identify graspable areas is essential for the success of this research. This study introduces a novel system for detecting grasp regions in household objects using a network based on part segmentation. We have compiled a comprehensive database of household items, meticulously segmenting them into graspable and non-graspable components. Our criteria for graspability are informed by the typical usage of these tools; for example, the handle of a knife is considered graspable, as it is the part typically grasped when moving the knife. We utilize an enhanced DGCNN [20] architecture for part segmentation, applying it to our dataset by modifying the spatial transform block and addressing label imbalance within the loss function. The main contribution of our work can be summarized as follows :

- Development of a DGCNN-based Grasp Detection System: This paper presents a novel system for detecting graspable regions in household objects, leveraging the DGCNN network for part segmentation to identify meaningful object components essential for robotic manipulation.
- Compilation of a Comprehensive Database. A dedicated database of household objects has been compiled, systematically segmented into graspable and non-graspable parts based on their functional usage (e.g., categorizing handles as graspable).
- Enhancement of the DGCNN Architecture. The standard DGCNN has been improved by modifying the spatial transform block and selecting a suitable loss function to address class imbalance, making it better suited for the grasp detection task.

In the following sections, we first review the literature in Section 2, focusing on two key areas: computer vision-based robotic grasping and point cloud segmentation. In Section 3, we present our proposed method and introduce our database. Section 4 evaluates the results, and in Section 5, we discuss the findings in more detail, including suggestions for potential alternative grasp scenarios. Finally, Section 6 provides the conclusion of our work.

2 Literature Review

2.1 Vision-based robotic grasping

In 2020, Monica and Aleotti [17] introduced a projective analysis method for part-based grasp planning. They employed the Bi-class Symmetric Hausdorff distance (BiSH) for point cloud categorization, transferring labels from pre-labeled images to 2D point cloud projections, which were subsequently fused back onto the 3D point cloud for part segmentation. Pose estimation for fruit grasp planning [21] involves point cloud registration in two stages: first, coarse registration is performed using the Sample Consensus Initial Alignment (SAC-IA) technique to align the laser scanner template with the Kinect point cloud, followed by fine registration with the Iterative Closest Point (ICP) algorithm to refine and finalize the 6D pose. Grasp planning using superimposed segmentation [3] involves the use of computer-aided design (CAD) models of target objects to develop grasp configurations. The core technique of this approach, superimposed segmentation, preprocesses the mesh model by dividing it into overlapping facets. In 2022, Etxezarreta and Sagardia [22] developed a real-time application for predicting hand-object contact in complex geometries. Their method simplifies the Voxelmap-Pointshell collision detection system, handling resolution constraints and supporting interactions between multiple objects using point clouds and voxelized signed distance fields. Graspability map generation [23] combines data-driven grasp planning

with analytical quality guarantees by creating high-resolution maps stored in a database. During runtime, customized maps are generated based on the properties of the gripper and the object to be grasped.

Working with multi-fingered robotic hands presents significant challenges due to the high-dimensional search space; however, they offer a more stable grasp across a diverse range of objects [24], [25]. Fan and Tomizuka [24] proposed a framework that incorporates the multi-dimensional iterative surface fitting (MDISF) algorithm for grasp planning, which aligns the hand surface with the object while minimizing collisions and surface fitting errors. Furthermore, the framework includes the grasp trajectory optimization (GTO) algorithm, which plans finger trajectories for grasp execution based on the object's point cloud representation. Marios Kiatos et al. [25] introduced a shape complementarity metric for multi-fingered hands, which features a fast algorithm for generating potential collision-free grasps in cluttered environments. Their approach employs an objective function to assess the shape complementarity between the hand and the object, effectively enhancing grasp stability and efficiency in complex scenarios. In 2021, Lundell et al. [26] introduced Deep Dexterous Grasping in Clutter (DDGC) to accelerate the generation of high-quality, collision-free multi-finger grasps. DDGC accomplishes this by integrating scene completion, scene encoding, and a differentiable forward kinematics layer, enabling efficient handling of unknown objects in cluttered environments.

In 2020, Zhang et al. [19] proposed a grasp saliency map method to predict critical grasp points. They introduced a technique for transferring saliency maps across different shapes within the same class by leveraging feature extraction and correspondence. Building on this, they developed a deep neural network inspired by PCPNet, which combines global feature guidance with local patch inputs to estimate grasp saliency. In 2020, Qian et al. [16] introduced a task-constrained grasp pose detection method for single-view point clouds, utilizing a convolutional neural network for pixel-level affordance detection. They enhanced grasp accuracy and stability by refining the local frame calculation in the Grasp Pose Detection (GPD) method and implementing a position-sensitive fully convolutional neural network for grasp stability classification. Wang et al. [27] developed a hierarchical policy algorithm in 2022 that utilizes latent trajectory embeddings for effective planning. The high-level policy generates these embeddings from a partially observed point cloud, while a Q-learning-trained critic network scores them. The low-level policy subsequently selects the highest-scored embedding to produce a sequence of end-effector poses for the robot. Liu et al. [7] proposed a grasp detection method utilizing point clouds to train a PointNet++-based network, improving feature capture of grasped samples. They employed an antipodal-based sampling scheme from Dex-Net and evaluated grasp quality using metrics that incorporate both force closure and epsilon quality.

Yan et al. [12] proposed a lightweight RGB-D fusion module named self-attention-based multi-scale confidence map fusion (SMCF), which effectively merges the multi-scale confidence map with a self-attention mechanism. They also developed the Attention Feature Fusion (AFF) module to adaptively combine the segmentation output with features from the grasp detection network, along with the Feature Fusion Atrous Spatial Pyramid (FFASP) to address challenges in generating grasp candidates for small objects. Wan et al. [28] introduced an instance segmentation method in 2023 using the Instance-Augmented Net (IAN) pipeline, which integrates instance information into feature extractors. This method effectively addresses excessive noisy features resulting from misidentifying points belonging to the same object, while also enhancing the utilization of multi-resolution information in instance segmentation backbones. The Cross-Window Point Transformer (CP-Former) [18] enhances 3D object segmentation from incomplete single-view data by employing a bidirectional cross-attention mechanism that captures long-range dependencies and refines point-wise features. It improves segmentation performance by emphasizing latent boundary regions through contrastive learning and an adaptive dual aggregation strategy.

2.2 Point cloud segmentation

One of the key advancements in point cloud processing is the introduction of PointNet [29], the first neural network architecture specifically designed for the direct processing of raw point cloud data. The network comprises three main components: max pooling layers, a multi-layer perceptron (MLP) module, and feature representation. The max pooling structure addresses the unordered nature of point cloud data, enabling PointNet to achieve permutation invariance. As a global feature representation, the max pooling layer functions as a symmetric operator, aggregating information by selecting the maximum feature value from each set of points. The MLP module serves as a classifier for the global features of the shape and facilitates the extraction of point cloud features through weight sharing. Additionally, the feature fusion structure integrates both local and global information by concatenating the global point cloud feature vector with each point feature, thereby updating per-point features to include both types of information. To simplify the learning of an effective

rotation matrix, PointNet also incorporates the T-Net structure. Overall, PointNet remains a pivotal network design in this field, demonstrating success in tasks such as object categorization and semantic segmentation [20], [29].

Despite its significant strengths, PointNet's pointwise approach, which ignores spatial relationships between points, limits its ability to capture local geometric features. To address this limitation, PointNet++ was introduced in 2017 by Qi et al. [30]. Similar to CNNs, PointNet++ analyzes a set of points sampled hierarchically within a metric space. It achieves this by extracting local features from small regions, effectively capturing subtle geometric patterns. Within PointNet++'s architecture, PointNet has been selected as the local feature learner. These local features are progressively combined into larger groupings until the characteristics of the entire point set are captured. This approach enables the extraction of multi-scale features from point clouds [29], [30].

In 2019, Wang et al. [20] proposed the DGCNN, a novel method for point cloud processing that incorporates the EdgeConv module as a core component. EdgeConv generates edge features that describe the relationships between a point and its neighbors by constructing a graph through k-nearest neighbors. Unlike traditional Graph CNNs, where the graph remains fixed, DGCNN features a dynamically updated graph that evolves after each layer of the network, allowing for more flexible and adaptive feature learning. By stacking multiple layers, the network can progressively learn the global shape properties of an object, integrating local information at each layer to capture the overall structure. In multi-layer systems, points that are closer in feature space are recognized as having similar semantic characteristics, even if they are distant in the original space, enabling the network to uncover hidden features across larger distances. These capabilities make DGCNN particularly useful for tasks such as classification, semantic segmentation, and part segmentation [20].

3 Methodology

Our grasp region detection algorithm is built upon the DGCNN network, which, as previously mentioned, is a powerful tool for part segmentation. First, we will provide a detailed description of the method and network architecture, followed by an explanation of the data creation process. Finally, we will discuss the selected loss function for this task.

3.1 Graph formation and edge convolution

Consider a 3D point cloud consisting of n points, denoted as $P = \{X_1, X_2, \dots, X_n\} \subseteq R^3$, where each point X_i has coordinates $X_i = (x_i, y_i, z_i)$. A directed graph can be defined as $\mathcal{G} = (\mathcal{V}, \varepsilon)$, where $\mathcal{V} = \{1,2, \dots, n\}$ represents the set of vertices, and $\varepsilon \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of directed edges between these vertices. To convert a point cloud into a directed graph, in addition to treating the point set P as the vertices of the graph, we also need to define the edges. One of the simplest ways to define edges in a point cloud is by using the Euclidean distance between points in combination with a k-nearest neighbors (k-NN) approach.

A key component in the DGCNN architecture is the edge feature, which plays a critical role in capturing relationships between points in the point cloud. The edge feature $e_{ij} = h_\Theta(X_i, X_j)$, connecting two points X_i and X_j, is parameterized by Θ, which is learned during training. The formula for the edge feature, which combines the general information of X_i and the local information from the relative difference $X_j - X_i$, is expressed as follows:

$$h_\Theta(X_i, X_j) = \bar{h}_\Theta(X_i, X_j - X_i) \quad (1)$$

In practice, the edge feature h_Θ is implemented using MLP layers, which facilitate effective learning of complex relationships between the vertex features and their local differences. Based on this explanation, the final notation of the edge features can be expressed as follows:

$$e'_{ijm} = RELU(\theta_m . (X_j - X_i) + \phi_m . X_i), \quad (2)$$

Where $\Theta = (\theta_1, \dots, \theta_M, \phi_1, \dots, \phi_M)$, with M representing the number of different filters. Finally, the EdgeConv module defines the features of the graph points by applying a max aggregation function, as illustrated below:

$$x'_{im} = \max_{j:(i,j)\in\varepsilon} e'_{ijm} \quad (3)$$

A schematic of the EdgeConv module is shown in Figure 1. In this module, a tensor of size $n \times f$ (where f represents the number of point-wise features, which varies across layers) is processed by forming a graph and calculating k edge features using an MLP network with $\{a_1, a_2, \dots, a_n\}$ neurons. This results in a tensor of size $n \times k \times a_n$. After applying pooling, the output tensor will have a size of $n \times a_n$, representing a_n features for each point. One of the unique attributes of the DGCNN method is the dynamic generation of the graph at each layer. In the first layer, the EdgeConv module is used to construct the graph based on the nearest neighbors in Euclidean space. In subsequent layers, the graph is formed based on the proximity of vertices in the feature space derived from the previous layer, allowing for a more meaningful representation of relationships between points.

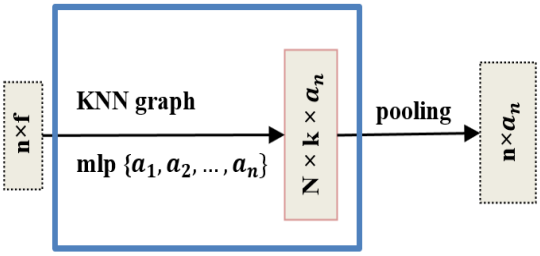

Fig. 1 Schematic of EdgeConv module.

3.2 Network architecture

The network architecture is illustrated in Figure 2. Our architecture closely resembles the DGCNN framework, with a minor modification pertaining to the spatial transform section of the network. A 3×3 matrix is commonly estimated by spatial transform blocks to align an input point set with a canonical space. In the DGCNN method, the k-nearest neighbors' algorithm is used to identify neighboring vertices. Subsequently, a tensor of size $n \times k \times 6$ is constructed using these vertices along with their differences from their neighbors. Through neural network learning, a transformation matrix is generated, enabling the effective transformation of the point cloud. Notably, using k-NN in Euclidean space to find neighborhoods presents challenges, such as non-uniform density. In regions of the point cloud where point density varies, k-NN may inadvertently connect distant points in sparse areas or fail to capture nearby connections in denser regions, leading to inaccurate neighborhood representations. To address these challenges, we replaced k-NN with a meshing-based method for extracting point neighbors, differing from DGCNN. Specifically, we utilized the Point2Mesh [31] technique to reconstruct a surface mesh from the input point cloud, leveraging a self-prior automatically derived from the point cloud itself. This approach yields more reliable and accurate neighborhood connections.

As illustrated in Figure 2, the segmentation model processes an input consisting of n points representing an object. Initially, the points undergo a modified spatial transformation. Following this transformation, the transformed points are sequentially fed into three EdgeConv modules, each performing feature extraction. An MLP network, combined with max pooling, is employed to generate a 1D global descriptor with 1024 hidden dimensions. This process aggregates the features extracted from the point cloud, allowing for a comprehensive representation that captures the most significant information across all points. Additionally, a categorical vector is employed during both the training and testing phases to manage the various classes of objects, ensuring that the network can distinguish between different object categories. This vector serves as an additional input feature, providing explicit information about the object class (e.g., bottle, mug, or scissors). By doing so, the network focuses on the relevant segmentation labels (graspable and non-graspable parts) specific to the given class. Incorporating this categorical vector prevents confusion between segmentation labels of different classes and improves the accuracy of predictions for each object type. Finally, the model generates per-point classification scores for p semantic labels by concatenating the 1D global descriptor with the outputs of all EdgeConv module, which serve as local descriptors for each point. In this study, we utilize this network architecture to transform the problem of detecting graspable regions into a part segmentation task by dividing each object class into two components: a graspable part and a non-graspable part.

3.3 Database creation

With the understanding that everyday objects are typically grasped from specific parts for task-oriented purposes, we have developed a 3D point cloud database for these objects. This database aims to enhance robotic grasping by providing representations of grasping locations suitable for a moving task. It includes six classes: bottles, eyeglasses, headphones, knives, mugs, and scissors. We collected approximately 1,120 3D point clouds from these six classes and manually segmented them into graspable and non-graspable parts using MeshLab software [33]. The graspable parts were carefully selected to ensure they are suitable for carrying the objects.

3.4 Loss function

In part segmentation tasks, some classes may experience imbalance. For example, when a bottle is divided into two parts (the cap and the body), the cap contains significantly fewer points than the body. To address this issue, we apply weighted softmax cross-entropy to ensure that both parts are learned effectively, compensating for the uneven distribution of points. This can improve the model's performance for underrepresented classes and help balance the contribution of each class to the overall loss. A weight score vector, derived from the distribution of labeled points in the training set, is incorporated into the loss function. As each object is divided into two parts, the vector consists of two weights. These weights are computed using the following equations, as described in [32].

$$D_i = \frac{n_1+n_2}{n_i} \tag{4}$$

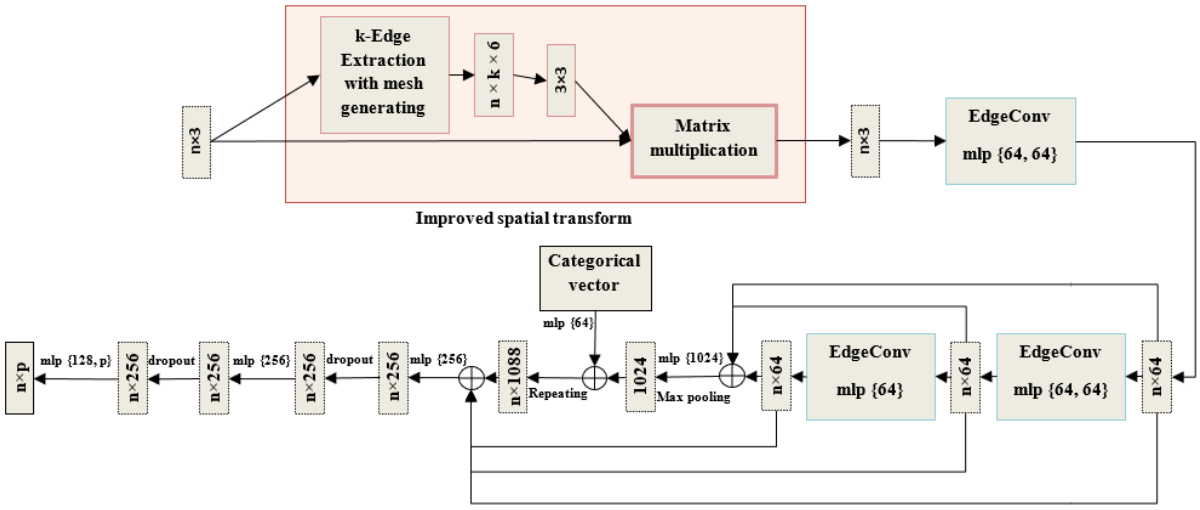

Fig. 2 Overview of the network architecture.

Let D_i represent the distribution of points labeled as i, where n_i is the number of samples labeled as i. The weight score for segment i is then obtained as follows:

$$\text{weight score}_i = \frac{D_i}{D_1+D_2} \tag{5}$$

4 Evaluation

In this section, we start by preparing the data and configuring the network architecture. Subsequently, we report numerical results using the IoU metric and compare the proposed method with benchmark approaches. Following this, we perform an ANOVA test and conduct a T-test to further assess the superiority of our method. Finally, we present visual results to demonstrate the effectiveness of our approach.

4.1 Data

We test our DGCNN-based model on a prepared database consisting of 3D point clouds from six classes, each divided into two parts, resulting in a total of 12 segments. To enhance the quality of training, we augmented the data by adding noise and applying rotation and translation. For each shape in the dataset, 2,048 points are sampled. The data is divided into three parts: training, validation, and testing, with a split ratio of 70%, 15%, and 15%, respectively.

4.2 Architecture

The network architecture is detailed in Section 3.2, where we applied a dropout rate of 0.5 to two MLP layers, each containing 256 units. Additionally, the number of neighbors used in the EdgeConv modules is set to 20. The input to our network consists of the 3D coordinates of the point cloud, represented as an $n \times 3$ matrix, where n denotes the number of points. For our experiments, we sampled 2048 points from each shape in the dataset, resulting in a 2048×3 input to the network. The output of the network is an 2048×12 matrix, where each point is assigned one of 12 segmentation labels.

4.3 Numerical results

Intersection-over-Union (IoU) on points is a metric used to evaluate our model and compare its performance with other benchmarks. Both the network's output and the ground truth data have identical dimensions, allowing for a direct comparison during the IoU calculation. The IoU for each segment is computed individually using the formulas (6) and (7):

$$\text{IOU}_1 = \frac{|A_1 \cap B_1|}{|A_1 \cup B_1|} \tag{6}$$

$$\text{IOU}_2 = \frac{|A_2 \cap B_2|}{|A_2 \cup B_2|} \tag{7}$$

Where A_1 and A_2 represent the ground truth points for segment 1 (graspable part) and segment 2 (non-graspable part), respectively, and B_1 and B_2 represent the predicted points for segment 1 and segment 2, respectively. The final IoU for the entire object is computed as follows:

$$IoU_{Object} = \frac{IOU_1 + IOU_2}{2} \tag{8}$$

The IoU for a class is calculated by averaging the IoUs of all objects within that class and named mean IoU (mIoU) and then the IoUs of all classes are averaged to determine the grand mean IoU (gmIoU). The table 1 shows the mIoU and gmIoU results for the proposed method, as well as the PointNet and PointNet++ methods.

Table 1 Part segmentation results on the household dataset. The metrics used are mIoU and gmIoU in percentage, calculated on points.

Method	gmIoU	mIoU					
		Bottle	Eyeglasses	Headphones	Knife	Mug	Scissors
Pointnet	87.22	87.1	90.8	89.21	88.1	93.01	75.1
Poinnet++	87.77	88.5	91.2	88.01	88.2	94.2	76.5
Ours (Without Weighted Loss)	91.73	91.3	95.6	92.42	92.5	95.01	83.58
Ours (Weighted Loss)	**92.57**	**93.54**	**95.6**	**92.73**	**92.6**	**97.09**	**83.7**

As can be seen, our method achieves higher mIoU and gmIoU compared to the other methods. We also present the mIoUs of the method before and after applying the weighted loss function. As seen, the results for the bottle and mug classes increased by 2.24% and 2.08%, respectively, after applying the weighted loss function. The results for the other classes did not change significantly due to the better balance between the segments. Additionally, the gmIoU increased by 0.84% after applying the weighted loss function.

Figure 3 (a–c) presents box plots of the mIoU for PointNet, PointNet++, and the proposed method across six distinct object classes: bottle, eyeglasses, headphones, knife, cup, and scissors. Each box plot visualizes the distribution of mIoU values for the corresponding class, displaying key statistical measures such as the maximum, minimum, median, and the upper and lower quartiles of the data. A comparison of these box plots shows that the distribution range is narrower for our method across all classes. In Figure 3 (d), box plots of the gmIoU for the enhanced DGCNN network, PointNet, and PointNet++ are presented. The results for PointNet and PointNet++ show a wider distribution range in gmIoU, indicating that their performance is more sensitive to variations in test samples. In contrast, the results of our proposed method exhibit a more concentrated distribution, highlighting superior consistency compared to the other methods.

4.4 Statistical analysis

In this section, we evaluate the performance of the proposed method using statistical tools, including T-tests and ANOVA. The T-test compares the means of two groups to determine if the observed differences are statistically significant, while ANOVA extends this analysis to multiple groups by evaluating variations within and between them. Table 2 presents the unpaired T-test results comparing the Enhanced DGCNN method with PointNet, while Table 3 presents the results comparing the Enhanced DGCNN with PointNet++. The Enhanced DGCNN outperforms both PointNet and PointNet++ significantly, as shown in the T-test results in Table 2 and Table 3. Compared to PointNet (Table 2), the Enhanced DGCNN achieves a mean difference of 0.05349 ($p < 0.0001$), with a 95% confidence interval ranging from 0.04103 to 0.06595. Similarly, when compared to PointNet++ (Table 3), the Enhanced DGCNN demonstrates a mean difference of 0.04799 ($p < 0.0001$), with a 95% confidence interval of 0.03588 to 0.06011.

The F-test for variance comparison further supports the consistency of the Enhanced DGCNN's results, with significant variance differences observed for both PointNet ($F = 2.560$, $p < 0.0001$) and PointNet++ ($F = 2.364$, $p < 0.0001$). These findings indicate that the Enhanced DGCNN not only provides better performance in terms of mean accuracy but also exhibits greater stability across datasets.

The ANOVA test shows a significant F-value of 33.67 ($p < 0.0001$), indicating that there are statistically significant differences between the methods, suggesting that the mean performance of the three methods differs significantly. Since the ANOVA test showed significant differences among the three methods, pairwise comparisons were conducted using Dunnett's test, as presented in Table 4.

The results demonstrate that the proposed method significantly outperforms both PointNet and PointNet++: the mean difference compared to PointNet is 0.05349 ($P < 0.0001$), and the mean difference compared to PointNet++ is 0.04799 ($P < 0.0001$). These findings further confirm the superior performance of the proposed method.

Table 5 summarizes the results of the T-test and ANOVA comparing the proposed method with PointNet and PointNet++, demonstrating that, statistically, the proposed method outperforms both PointNet and PointNet++.

4.5 Visual results

We also present the visual results of our method alongside the ground truth for better clarity in Figure 3, which includes six categories of shapes. In Figure 4, we observe examples of noisy shapes from these six categories that have been segmented using the proposed method, demonstrating the effectiveness of the approach in the presence of low noise.

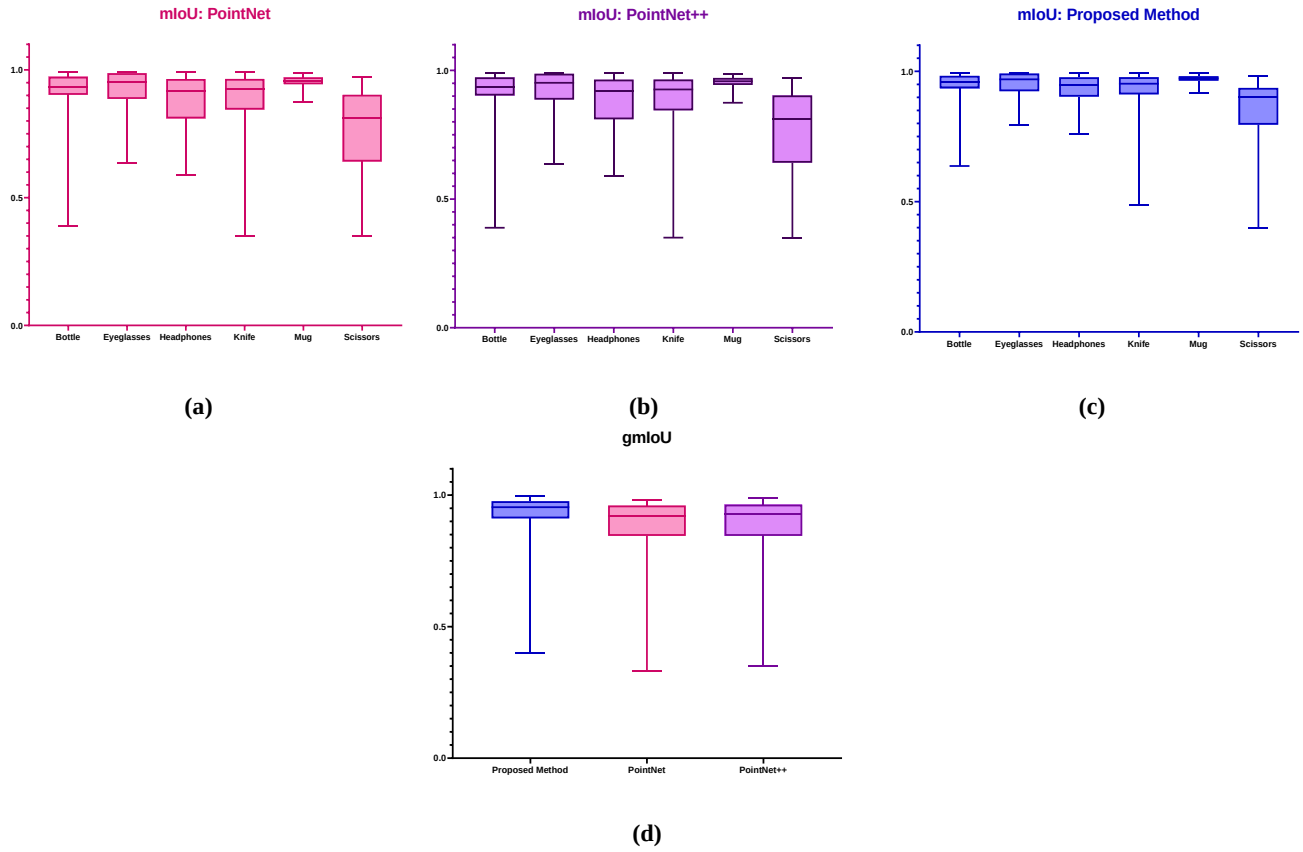

Fig. 3 Box plots of the mIoU and gmIoU for three different methods. (a–c) Box plots depicting the mIoUs for PointNet, PointNet++, and the proposed method across six distinct object classes. (d) Box plots compare the gmIoU of PointNet, PointNet++, and the proposed method.

5 Discussion

The appropriate grasping location of objects varies across different tasks and scenarios, and in some cases, multiple grasping regions can be identified for a single object. In our work, we focused on regions suitable for carrying objects, leveraging the human tendency to grasp narrow and small areas, such as the handle or cap of a mug or bottle, when selecting the graspable regions. A potential solution for other tasks could involve defining task-specific graspable regions and training the network with these new regions. Additionally, different graspable parts could be incorporated for various tasks using a multi-task segmentation approach instead of binary segmentation.

6 Conclusion

In this work, we proposed an algorithm for identifying the grasp locations of objects, utilizing the DGCNN network to segment and detect the graspable parts. The DGCNN architecture was modified and used for a custom database specifically collected for this research. Our results have outperformed the discussed methods, demonstrating superior mIoU and gmIoU scores, along with achieving better performance confirmed by statistical methods. Furthermore, our approach has shown strong performance against noisy data, thanks to training on datasets augmented with noise, rotation, and translation. Future work will focus on expanding the database to further enhance the model's performance, including more object categories, and utilizing other networks to improve grasp area detection.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Haniye Merrikhi: Conceptualization, Methodology, Formal analysis, Coding and Execution, Writing - Original draft.

Prof. Dr. Hossein Ebrahimnezhad: Supervision, Revise & editing, Investigation.

Funding

No funding was received for this work.

Informed Consent Statement

Not applicable.

Table 2 T-Test Results Comparing the Enhanced DGCNN Method with PointNet.

Table Analyzed	**Unpaired t test data**
Column B	Proposed Method
vs.	vs.
Column A	PointNet
Unpaired t test	
P value	<0.0001
P value summary	****
Significantly different (P < 0.05)?	Yes
One- or two-tailed P value?	Two-tailed
t, df	t=8.421, df=1300
How big is the difference?	
Mean of column A	0.8722
Mean of column B	0.9257
Difference between means (B - A) ± SEM	0.05349 ± 0.006352
95% confidence interval	0.04103 to 0.06595
R squared (eta squared)	0.05173
F test to compare variances	
F, DFn, Dfd	2.560, 650, 650
P value	<0.0001
P value summary	****
Significantly different (P < 0.05)?	Yes
Data analyzed	
Sample size, column A	651
Sample size, column B	651

Table 3 T-Test Results Comparing the Enhanced DGCNN Method with PointNet++.

Table Analyzed	**Unpaired t test data**
Column B	Proposed Method
vs.	vs.
Column A	PointNet++
Unpaired t test	
P value	<0.0001
P value summary	****
Significantly different (P < 0.05)?	Yes
One- or two-tailed P value?	Two-tailed
t, df	t=7.773, df=1300
How big is the difference?	
Mean of column A	0.8777
Mean of column B	0.9257
Difference between means (B - A) ± SEM	0.04799 ± 0.006174
95% confidence interval	0.03588 to 0.06011
R squared (eta squared)	0.04441
F test to compare variances	
F, DFn, Dfd	2.364, 650, 650
P value	<0.0001
P value summary	****
Significantly different (P < 0.05)?	Yes
Data analyzed	
Sample size, column A	651
Sample size, column B	651

Table 4 Results of ANOVA and Dunnett's Multiple Comparisons Test for Pairwise Comparisons Between the Proposed Method, PointNet, and PointNet++.

Number of families	1
Number of comparisons per family	2
Alpha	0.05

Dunnett's multiple comparisons test	**Mean Diff.**	**95.00% CI of diff.**	**Significant?**	**Summary**	**Adjusted P Value**	**A-?**	
Proposed Method vs. PointNet	0.05349	0.03868 to 0.06830	Yes	****	<0.0001	B	PointNet
Proposed Method vs. PointNet++	0.04799	0.03318 to 0.06280	Yes	****	<0.0001	C	PointNet++

Test details	**Mean 1**	**Mean 2**	**Mean Diff.**	**SE of diff.**	**n1**	**n2**	**q**	**DF**
Proposed Method vs. PointNet	0.9257	0.8722	0.05349	0.006690	651	651	7.995	1950
Proposed Method vs. PointNet++	0.9257	0.8777	0.04799	0.006690	651	651	7.174	1950

Table 5 ANOVA test and T-test on metrics (gmIoU) for different segmentation methods

	Proposed Method vs Point Net		**Proposed Method vs Point Net++**	
p Value	ANOVA	T-Test	ANOVA	T-Test
MIoU	P < 0.0001	P < 0.0001	P < 0.0001	P < 0.0001

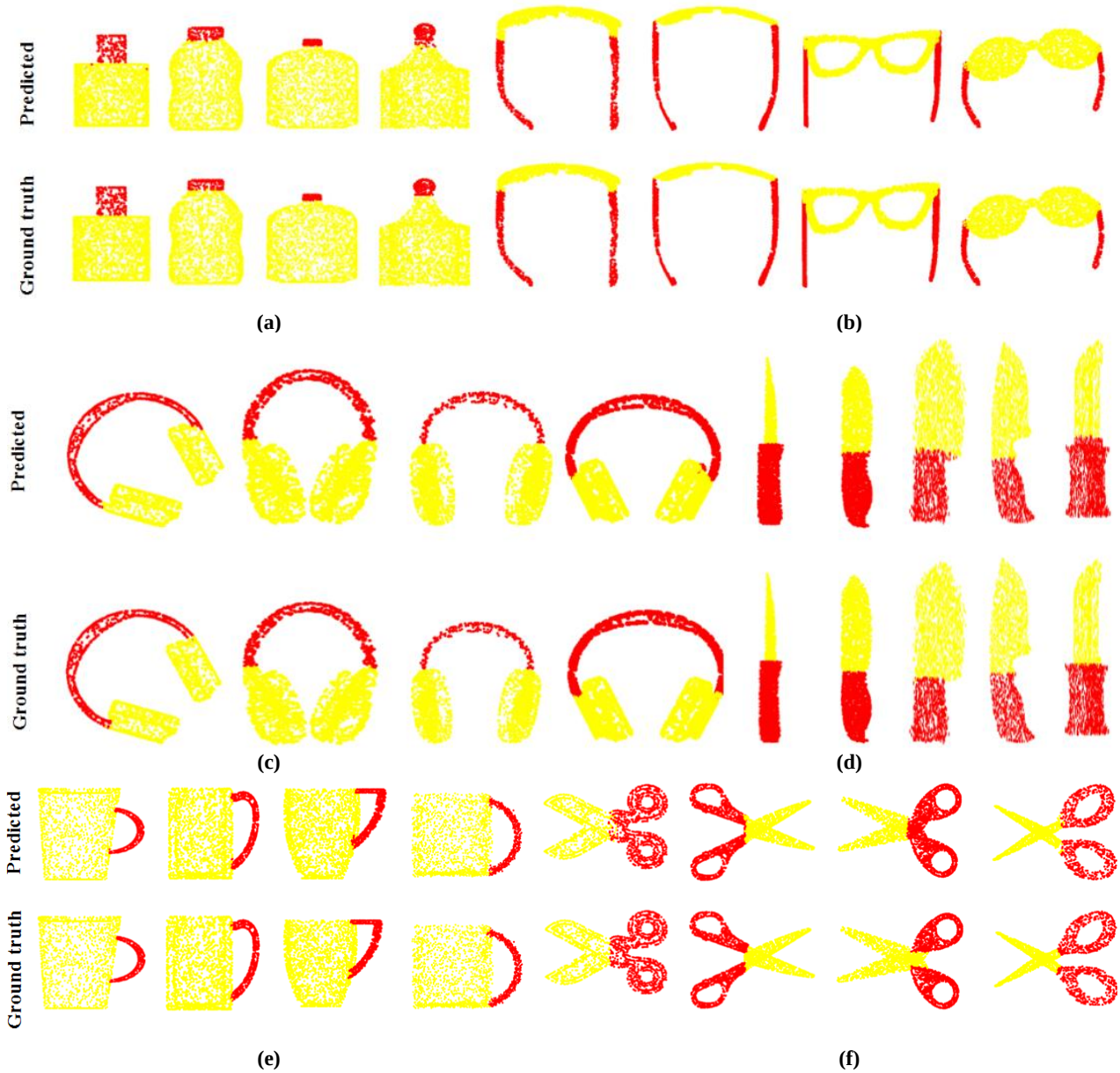

Fig. 4 Visual results of graspable part detection. The figure illustrates the segmentation of household objects into graspable (red) and non-graspable (yellow) parts. (a) corresponds to a bottle, (b) to eyeglasses, (c) to headphones, (d) to a knife, (e) to a mug, and (f) to scissors.

Fig. 5 Graphical outcomes for noisy data

References

[1] N. Guo, B. Zhang, J. Zhou, K. Zhan, and S. Lai, "Pose estimation and adaptable grasp configuration with point cloud registration and geometry understanding for fruit grasp planning," *Comput. Electron. Agric.*, vol. 179, 2020.

[2] J. Rong, P. Wang, T. Wang, L. Hu, and T. Yuan, "Fruit pose recognition and directional orderly grasping strategies for tomato harvesting robots," *Comput. Electron. Agric.*, vol. 202, no. September, p. 107430, 2022.

[3] W. Wan, K. Harada, and F. Kanehiro, "Planning Grasps with Suction Cups and Parallel Grippers Using Superimposed Segmentation of Object Meshes," *IEEE Trans. Robot.*, vol. 37, no. 1, pp. 166–184, 2021.

[4] F. J. Chu, R. Xu, and P. A. Vela, "Real-world multiobject, multigrasp detection," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3355–3362, 2018.

[5] X. Zheng, J. Rong, Z. Zhang, Y. Yang, W. Li, and T. Yuan, "Fruit growing direction recognition and nesting grasping strategies for tomato harvesting robots," *J. F. Robot.*, vol. 41, no. 2, pp. 300–313, 2024.

[6] J. Deng, H. Zhang, J. Hu, X. Zhang, and Y. Wang, "Class Incremental Robotic Pick-and-Place via Incremental Few-Shot Object Detection," *IEEE Robot. Autom. Lett.*, vol. 8, no. 9, pp. 5974–5981, 2023.

[7] X. Liu, C. Huang, J. Li, W. Wan, and C. Yang, "Two-Stage Grasp Detection Method for Robotics Using Point Clouds and Deep Hierarchical Feature Learning Network," *IEEE Trans. Cogn. Dev. Syst.*, vol. 16, no. 2, pp. 720–731, 2024.

[8] G. Du, K. Wang, S. Lian, and K. Zhao, *Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review*, vol. 54, no. 3. Springer Netherlands, 2021.

[9] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Rob. Auton. Syst.*, vol. 60, no. 3, pp. 326–336, 2012.

[10] K. Samarawickrama, "Kulunu Samarawickrama RGBD BASED DEEP LEARNING METHODS," no. May, 2021.

[11] I. Sarantopoulos and Z. Doulgeri, "Human-inspired robotic grasping of flat objects," *Rob. Auton. Syst.*, vol. 108, pp. 179–191, 2018.

[12] Y. Yan, L. Tong, K. Song, H. Tian, Y. Man, and W. Yang, "SISG-Net: Simultaneous instance segmentation and grasp detection for robot grasp in clutter," *Adv. Eng. Informatics*, vol. 58, no. May, p. 102189, 2023.

[13] D. Eizicovits and S. Berman, "Automatic graspability map generation based on shape-primitives for unknown and familiar objects," *Rob. Auton. Syst.*, vol. 110, pp. 1–11, 2018.

[14] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A Survey on Learning-Based Robotic Grasping," *Curr. Robot. Reports*, vol. 1, no. 4, pp. 239–249, 2020.

[15] L. Antanas *et al.*, "Semantic and geometric reasoning for robotic grasping: a probabilistic logic approach," *Auton. Robots*, vol. 43, no. 6, pp. 1393–1418, 2019.

[16] K. Qian *et al.*, "Grasp Pose Detection with Affordance-based Task Constraint Learning in Single-view Point Clouds," *J. Intell. Robot. Syst. Theory Appl.*, vol. 100, no. 1, pp. 145–163, 2020.

[17] R. Monica and J. Aleotti, "Point Cloud Projective Analysis for Part-Based Grasp Planning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4695–4702, 2020.

[18] Q. Wang, D. Liu, Z. Liu, J. Xu, and J. Tan, "3D Object Segmentation Using Cross-Window Point Transformer with Latent Semantic Boundary Guidance," *IEEE Trans. Multimed.*, vol. 26, pp. 5951–5961, 2024.

[19] L. na Zhang, S. yao Wang, J. Zhou, J. Liu, and C. gang Zhu, "3D grasp saliency analysis via deep shape correspondence," *Comput. Aided Geom. Des.*, vol. 81, p. 101901, 2020.

[20] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph Cnn for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, 2019.

[21] N. Guo, B. Zhang, J. Zhou, K. Zhan, and S. Lai, "Pose estimation and adaptable grasp configuration with point cloud registration and geometry unde rstanding for fruit grasp planning," *Comput. Electron. Agric.*, vol. 179, p. 105818, 2020.

[22] A. Etxezarreta and M. Sagardia, "Real time contact surface prediction for grasping with complex geometries," *Comput. Graph.*, vol. 107, pp. 66–72, 2022.

[23] D. Eizicovits and S. Berman, "Automatic graspability map generation based on shape-primitives for unknown and familiar objects," vol. 110, pp. 1–11, 2018.

[24] Y. Fan and M. Tomizuka, "Efficient Grasp Planning and Execution With Multifingered Hands by Surface Fitting," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3995–4002, 2019.

[25] M. Kiatos, S. Malassiotis, and I. Sarantopoulos, "A Geometric Approach for Grasping Unknown Objects with Multifingered Hands," *IEEE Trans. Robot.*, vol. 37, no. 3, pp. 735–746, 2021.
[26] J. Lundell, F. Verdoja, and V. Kyrki, "DDGC: Generative Deep Dexterous Grasping in Clutter," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6899–6906, 2021.
[27] L. Wang, X. Meng, Y. Xiang, and D. Fox, "Hierarchical Policies for Cluttered-Scene Grasping with Latent Plans," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2883–2890, 2022.
[28] Z. Wan, J. Hu, H. Zhang, and Y. Wang, "IAN: Instance-Augmented Net for 3D Instance Segmentation," *IEEE Robot. Autom. Lett.*, vol. 8, no. 7, pp. 4354–4361, 2023.
[29] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-Janua, pp. 77–85.
[30] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
[31] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or, "Point2Mesh: A Self-Prior for Deformable Meshes," *ACM Trans. Graph.*, vol. 39, no. 4, 2020.
[32] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB 2020*, 2020, pp. 1–7.
[33] "MeshLab." https://www.meshlab.net/ (accessed Sep. 28, 2024).

Haniye Merrikhi was born in Tabriz/ Iran in 1991. She received her B.Sc. and M.Sc. degrees majoring in electrical engineering from Tabriz University and Sahand University of Technology in 2015 and 2018, respectively. Her main interest relies in computer vision, pattern recognition, and machine learning. Currently, she is a PhD student at Sahand University of Technology.

Hossein Ebrahimnezhad was born in Iran 1971. He received his B.Sc. and M.Sc. degrees in Electronic and Communication Engineering from Tabriz University and K.N. Toosi University of Technology in 1994 and 1996, respectively. In 2007, he received his Ph.D. degree from Tarbiat Modares University.
His research interests include image processing, computer Vision, 3D model processing and soft computing. Currently, he is a professor at Sahand University of Technology.