Spatial Feature Mapping for 6DoF Object Pose Estimation

Jianhan Mei^a, Xudong Jiang^a, Henghui Ding^{a,*}

^aSchool of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

Abstract

This work aims to estimate 6Dof (6D) object pose in background clutter. Considering the strong occlusion and background noise, we propose to utilize the spatial structure for better tackling this challenging task. Observing that the 3D mesh can be naturally abstracted by a graph, we build the graph using 3D points as vertices and mesh connections as edges. We construct the corresponding mapping from 2D image features to 3D points for filling the graph and fusion of the 2D and 3D features. Afterward, a Graph Convolutional Network (GCN) is applied to help the feature exchange among objects' points in 3D space. To address the problem of rotation symmetry ambiguity for objects, a spherical convolution is utilized and the spherical features are combined with the convolutional features that are mapped to the graph. Predefined 3D keypoints are voted and the 6DoF pose is obtained via the fitting optimization. Two scenarios of inference, one with the depth information and the other without it are discussed. Tested on the datasets of YCB-Video and LINEMOD, the experiments demonstrate the effectiveness of our proposed method.

Keywords: 6D Pose Estimation, Rotation Symmetry, Spherical Convolution, Graph Convolutional Network

^{*}Corresponding author

Email addresses: jianhan001@e.ntu.edu.sg (Jianhan Mei), exdjiang@ntu.edu.sg (Xudong Jiang), ding0093@ntu.edu.sg (Henghui Ding)

1. Introduction

In this work, we address the challenging task of estimating object 6D pose from the image. It aims at recovering the 6D pose of each object instance in an image. More specifically, we focus on the rigid object 6D pose that contains 3 rotation parameters (yaw, pitch, roll) and 3 translation parameters (x, y, d)along 3 axes in the 3D coordinate system.

Object 6D pose estimation is a fundamental computer vision task in many applications, *e.g*let@tokeneonedotrobot manipulation and autonomous driving. Like many other computer vision tasks, pose estimation faces typical challenges, such as occlusion among object instances, background clutter, dynamic changes in the environment. In recent years, the success of deep neural networks [1, 2, 3, 4, 5] in computer vision tasks has greatly promoted the development of pose estimation and has achieved many impressive achievements [6, 7, 8, 9, 10]. Regrettably, most methods either directly regress 6D parameters based on image features, or estimate poses from the corresponding keypoints for every single object. Therefore, the problem of rotation symmetry and the 3D positional relationship of objects' points still cannot be well addressed.



Figure 1: The main idea of our spherical correlation method is to learn a latent spherical feature representation for each object that is consistent with the object appearances under different views so that the rotation on the object leads to the same rotation on its spherical feature. Here the views of an object (in the first row) are indexed by the rotations (in the second row) on SO(3).

The challenge of the rotation symmetry comes from similar object image appearances under different object rotations, e.glet@tokeneonedota symmetric object labeled as different poses may have very similar or even the same image appearance. In the learning stage, such data and labels bring out the problem of one versus multiple mappings, which makes the direct regression of the pose parameter an ill-posed problem. This results in poses with similar appearances ambiguous. As illustrated in Fig. 1, the handle of the cup indicates the rotation of the cup. However, during changing the views, with the disappearance of the handle, the cup with different rotations will have the same appearance. This symmetry problem can be considered as a subset of the occlusion cases called self-occlusion, where the object occludes part of itself. Such pose ambiguity of symmetric objects may frequently occur in estimation, which is ill-posed problem as one versus multiple mappings is unsolvable by optimization. Efforts have been made on this symmetry problem. In [11], the causes of pose ambiguities are grouped into object self-symmetry and occlusion-induced symmetry. Motivated by the recent advance on spherical convolutional neural networks [12, 13], we propose to learn a latent spherical feature representation to object as an auxiliary for the pose estimation. The rotation parameter is on the Special Orthogonal (SO(3)) group which is consistent with the spherical correlation result so that the estimation of the rotation parameter can be proposed to be done by analyzing the spherical correlation defined in [12] between the object appearance feature and the learned sphere feature representation. The object appearance feature is mapped to a hemisphere representing its one-side view. In Fig. 1, the latent spherical representation is required to be with the same rotation property of the object. By finding the maximum correlation between the object appearance spherical feature and the latent spherical representation, we do not confuse the network learning to predict different rotation parameters from similar appearances. During the inference, we pick the rotation parameter that has the largest spherical correction, which does not have to be the groundtruth rotation parameters as long as the corresponding projection of the 3D model well aligns with the ground-truth projection.

Furthermore, in the camera coordinate system, traditional methods for estimating the object pose typically utilize hand-crafted features to find the correspondence between the camera image and a predefined 3D model. The pose parameters are then calculated according to the correspondence. Due to the common shortcomings of hand-crafted features (*i.e*let@tokeneonedot. SIFT [14]), such methods are very sensitive to the texture transition and lighting changes. Accordingly, the popularity of data-driven methods has brought new opportunities and challenges to pose estimation based on learning. More specifically, Deep Neural Networks (DNNs) have achieved remarkable success in many fields [15, 16, 17, 18, 19, 20, 21]. For example, [7, 22, 23, 24] propose that the pose parameters are directly regressed from the convolutional features of each object. In this case, such methods are going to achieve significant improvements if there are sufficient training samples. However, one of the main challenges of pose estimation is background clutter, where objects may be surrounded by complex backgrounds or occluded by other objects. Noise will be mixed into the object's convolutional features, which makes it difficult for the regressor to fit the data according to the feature representation. To overcome this problem, a method of obtaining translation from convolutional feature maps by voting is first proposed in PoseCNN [7].

Recently, inspired by the traditional methods of inferring pose parameters using the correspondence, [6, 25, 26] define and calculate the keypoints of the objects, and then find the pose based on the corresponding keypoints through the "Perspective-n-Point" (PnP) algorithm. Usually, the keypoints of these methods are obtained through a coordinate map according to the surface of the object in the image, and the keypoints are determined by voting of all pixels from the coordinate map. The advantage of this is not only that the keypoints prediction is pixel-by-pixel, which can handle occlusion and background noise in features, but also that voting effectively suppresses outliers. Following the above-mentioned pose estimation idea based on keypoints, [6, 25, 26, 27] predict the coordinate map directly from the 2D convolutional features where 3D information is insufficiently used. Moreover, [9, 28] merge the 2D features with 3D points [29], and the PointNet [30] is applied for further calculation. However, in most of the keypoint-based methods, the 3D relation information is not sufficiently considered. Although PointNet [30] is good at dealing with disordered points, the Multilayer Perceptron (MLP) may destroy the essential structure of the data. 6D object pose estimation is closely related to 3D geometry measurement. Among different coordinate spaces, the 6D pose parameter is one of the essential descriptions for objects in 3D space. There are a few ways to build the essential representation for a 3D object, e.glet@tokeneonedot, point cloud, voxel, mesh and 3D surface. While voxel, mesh and the 3D surface can easily store the 3D relationship between elements, the relation information is hard to be processed by neural networks. PointNet [30] and O-CNN [31] provide solutions for neural networks to deal with point cloud and voxel data respectively while Graph Convolutional Networks (GCNs) [32] make it possible for graph data inference by a neural network. While point cloud is leaking the 3D connection and voxel is redundant, mesh is considered a more elegant representation for a neural network with spatial relation information. Moreover, mesh is one kind of special graph that uses its nodes as vertices and connections as edges. Hence, instead of using point cloud or voxel, we connect discrete points into mesh to build a graph model of a 3D object. Next, GCN is used to help the feature exchange and merge. Finally, the predefined key points are regressed and voted from the vertices of the graph, and the pose parameters are calculated through optimization. Using the mesh representation for data inference provides the system with a nature graph scheme. Compared with MLP in PointNet [30], GCN considers more on the connection relationship and data spatial topology. Using GCN combined with the mesh representation sheds light on building a union 3D data representation for neural network inference.

Our contributions are summarized as follows:

• We explore to solve the rotation ambiguous and occlusion problems in 6D pose estimation by using the spherical correlation and graph convolution. A robust 6D object pose estimation system is proposed.

- We propose to map the 2D convolutional feature to both a sphere and the 3D mesh representation. And the corresponding network components are integrated which forms the end-to-end deep neural network training and inference scheme. The proposed method digs more essential information from the data representation and processing. To our knowledge, this is new and meaningful for 6D object pose estimation.
- Target on the rotation ambiguous and occlusion challenges, we propose to solve from the parameter space and data structure. The state-of-the-art performance demonstrates the efficiency of the proposed system.

2. Related Work

2.1. 6D Object Pose Estimation

Early methods focus on recovering poses by matching keypoints features between 3D models and images [33, 14, 34, 35, 36]. Correspondences are found through the matching and the parameters are recovered by further optimization. However, these methods are limited by hand-crafted features and suffer from the problems of the keypoints extraction and description, which are not robust to e.glet@tokeneonedotillumination changing, non-significant texture and affinetransform. Thus, with additional depth information, [37, 38, 39, 40, 41, 42]significantly improve the accuracy performance.

Recently, Deep Neural Networks (DNNs) have achieved remarkable success in many computer vision fields [43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]. The state-of-the-art 6D object pose estimation methods from monocular images are mostly based on DNNs. For example, convolutional features bring significant performance improvements [22, 56, 26]. More specifically, PoseCNN [7] proposes a representation that infers the actual 3D coordinates of the image coordinates then determines the object position center through Hough voting and designs ShapeMatch-Loss to solve the problem of rotationally symmetric object pose estimation. DeepIM [57] proposes a method to predict the relative translation and relative rotation of objects in two images so that the network can learn how to fine-tune the posture. In addition, it also proposes an untangled relative pose representation for accurate pose prediction. Unlike the pose estimation methods based on object detection, this representation can handle previously unseen objects. Besides, considering the 3D spatial properties of the object pose, more additional supervision information is used to enhance the performance of deep neural networks. [6, 25] learn the keypoints of 2D to 3D matching through the instance segmentation framework to enhance the description of the object pose, which brings the system performance to a new level. However, these methods all focus on utilizing the 2D features for better regression, where the 3D information may be insufficiently used. Thereupon, [9, 28] propose to do the fusion of 2D and 3D features. 2D points are sampled from the convolutional feature map. With the depth map, 3D coordinates are recovered from the sampled 2D points. The feature fusion is typically the concatenation of the 2D convolutional features and the 3D coordinates. Network inference in 3D space achieves better interpretation for the pose parameters and reaches more accuracy.

Note that [11, 58] specifically aim to address the symmetry ambiguous problem. Both of them try to learn a feature representation from multiple object views, and the final rotation is obtained by searching the feature space. Different from their methods, we propose to learn the object representation on a sphere which is an essential representation for object rotation and we do not require multi-view images.

2.2. Symmetric and Spherical Correlation

It is not difficult to understand that symmetric objects cause ambiguity in mapping appearance to pose. As aforementioned, the symmetric ambiguity makes the learning-based method into a one versus multiple mapping task, which is an ill-posed problem without a unique solution. By learning from such an ill-posed problem, the results fall into the unpredictable. Using such estimation for higher-level tasks may cause the system out of control. Recently, efforts have been made to tackle this problem. One intuitive solution is to do symmetry detection or analysis for each specific object. Efforts have been made on both extrinsic symmetry [59, 60, 61] and intrinsic symmetry [62, 63, 64, 65, 66] detection in 3D space [67]. However, such methods mainly focus on the symmetry of 3D models and often ignore the object image appearances under specific views, which are important for the pose parameter regression task.

In [22], a render and compare loss is introduced to compare the rotated and rendered 3D model with its segmentation and depth masks. Penalized by the Intersection over Union (IoU) between the rendered and the label masks, the ambiguous object poses are represented as hidden parameters in the network that can have the same appearance. However, solving the hidden parameters is still ill-posedd. Moreover, such render and compare loss only focuses on the rendering shape but ignores the object's texture. In [11], an encoder is learned that encodes the object poses in a feature codebook and the testing pose is obtained by retrieving from the codebook. However, learning the encoder needs training samples from all object rotation views. The rotation of an object on the SO(3) group is difficult to be uniformly sampled. In addition, the code distance for describing different rotation views is hard to define.

Spherical signals are a set of particular data with special properties on the sphere. It is known that the flat 2D convolution is a translation equivariance operation. Similarly, it is proved that the convolution on the sphere and SO(3) is equivariance on the rotation group $R \in SO(3)$ [12]. Particularly, in [12, 13], the spherical convolution on a sphere and SO(3) is defined and discussed. With that, we propose to learn a rotation equivariance spherical feature representation for 6D object pose estimation.

2.3. Graph Convolutional Network

GCN has demonstrated its strength in many fields. In 3D representation, the graph has been widely used. [68, 69] explore the application of GCN on the human body and hand joints, while [32] reconstruct the complete 3D mesh for the human hand using GCN. Considering that mesh and surface are popular 3D representations, they are all consisted of points and connections, which can be naturally abstracted as graphs. In [70, 71], the graph is built based on the 3D points and mesh connections, and GCN is used for the graph inference. The 3D shape generation is changed to the task of deformation from an initialized shape to the target shape. However, the inference of the graph is also a process of regression that predicts the target point coordinate on each node. From this view, [72] GCN is used to jointly learn the hand-object pose, where the object pose is described by its keypoints.

3. Methodology

As shown in Fig. 2, a monocular RGB image with its depth map is taken as the input of our overall pipeline, where the depth map is optional. The object masks and their Convolutional Neural Network (CNN) features are obtained in a semantic segmentation branch [73, 74]. A coarse rotation is learned by spherical learning. Inspired by [70], a graph is abstracted according to the object mesh formed by all the seen object surfaces. Afterward, the corresponding convolutional features combining the spherical feature are mapped to 3D coordinates and further refined through a graph matching algorithm. The Graph Convolutional Network (GCN) is used to process the graph model and a set of predefined 3D object keypoints are predicted. The graph unpooling layer is introduced so that the graph scale can be well controlled and the graph output can be dense enough to support the keypoint voting. The final pose parameters are obtained by optimization according to the keypoints.

3.1. Spherical Representation Learning

For the self-contained purpose, here we introduce some important concepts for spherical signals and operations on a sphere as well as on the SO(3) group introduced in [12, 13].

Without losing generality, we only introduce the continuous version of the spherical operations, which can be easily converted to the discrete case. Mainly following the notations in [12], similar to a 2D image, a sphere signal is defined



Figure 2: The overall pipeline of the proposed method. Our system is divided into four parts, which are a segmentation branch for feature extraction and sampling, a spherical learning branch for coarse rotation prediction, a GCN branch for capturing the 3D object structure and inferring the graph with 2D features, and a pose estimation utilizing the voted keypoints for parameter regression. The system takes an RGB image and a depth map as input, where the depth map is not compulsory. Our core data structure is a graph abstracted according to the 3D mesh connection. The graph is filled by 2D to 3D correspondence and filled with 2D convolutional features and spherical features. During the graph inference, unpooling will be applied so that the model can be better learned and there are enough output points to support the keypoint voting. In the GCN branch, two graph unpooling layers connect 3 groups of GCN layers. The posture parameters are calculated by optimizing the fitting function.

as a continuous function on the spherical coordinates $f : S^2 \to \mathbb{R}^K$, where S^2 indicates the spherical coordinates and K is the dimensionality of the continuous function. Rotating a spherical signal with a rotation operator O_R indexed by $R \in SO(3)$ has the property:

$$[O_R f](x) = f(R^{-1}x), (1)$$

where the R^{-1} is the inverse of R. (1) essentially states rotating a sphere signal by R is equivalent to rotating the sphere coordinate system by R^{-1} .

Considering that the normal 2D convolution can be defined as the function

of the inner product over the 2D plane, the inner product of two spherical signals is defined as:

$$[\phi * f] = \int_{S^2} \sum_{k=1}^{K} \phi_k(x) f_k(x) dx,$$
(2)

where ϕ is another spherical signal with the same dimensionality of f. Different from the inner product on the 2D plane, the inner product on sphere calculates the integration along sphere S^2 which is indexed by the spherical coordinates α and β . Its integration measure dx is a standard rotation invariant one, $d\alpha sin(\beta/4\pi)$, which ensures $\int_{S^2} f(Rx) dx = \int_{S^2} f(x) dx$ for any rotation $R \in SO(3)$. Further, the spherical correlation is defined as [12]:

$$[\phi * f](R) = \int_{S^2} \sum_{k=1}^{K} \phi_k(R^{-1}x) f_k(x) dx,$$
(3)

where $R \in SO(3)$ is the index of the spherical correlation.

The spherical correlation can also be defined on SO(3) [12]:

$$[\phi * f](R) = \int_{SO(3)} \sum_{k=1}^{K} \phi_k(R^{-1}Q) f_k(Q) dQ, \qquad (4)$$

where ϕ and f now denote functions defined on SO(3), and dQ denotes the invariant integration measure on SO(3). Under ZYZ-Euler angles, (α, β, γ) , dQ becomes $d\alpha sin(\beta)d\beta d\gamma/(8\pi^2)$ [12]. Both spherical correlations in (3) and (4) are rotation equivariance, *i.elet*@tokeneonedot.

$$[\phi * O_Q f](R) = [\phi * f](Q^{-1}R) = [O_Q[\phi * f]](R).$$
(5)

This property states that the result from a sphere correlation preserves the original signal property on SO(3).

This step aims to estimate coarse rotation parameters on SO(3). Our basic idea is to map the segmentation feature of an object onto a hemisphere. If there is a spherical representation of the object, we can use the spherical correlation to generate the correlation output on SO(3). Since we have the ground-truth 3D model for each object, a straightforward way is to follow [12] to use ray casting to map a 3D model to a sphere. Specifically, towards an origin, a ray shots from each point on the sphere surface and ends where it hits the model surface. The ray length and surface angle of each point are recorded and form the spherical signal. Using the ray casting sphere f as the reference spherical representation of the object, the object rotation can be found by learning the object feature under each specific view that has the maximum correlation with f. For a specific symmetric 3D model, when the ray origin coincides with its symmetry center, the ray casting spherical signal has the same symmetry attribute. However, the ambiguous poses come from both the object self symmetry and the occlusion induced symmetry. It is hard to find the symmetry origin for occlusion induced symmetry by only using the 3D model.



Figure 3: The segmentation feature is mainly learned from the ground-truth mask, and the spherical representation is mainly learned according to the ground-truth rotation.

Thus, instead of using the ray casting sphere f, we propose to learn an auxiliary spherical representation f that has the same symmetry property as the object, to handle the pose ambiguity. Specifically, the spherical representation f is represented by a branch of learned object-specific parameters of a multi-layer SphereCNN [12], where its input is the segmentation feature of an object projected to a hemisphere and its output is the spherical correlation on the angle grid sampled on the rotation angles on SO(3), as illustrated in Fig. 2.

In particular, we use the "Driscoll-Healy" grid following [75, 76] to project the segmentation feature of an object to a hemisphere denoted as ϕ . The spherical representation f or a branch of the SphereCNN consists of 5-layer parameters, where the first layer performs the spherical correlations on S^2 defined in Eq. 3 and the rest four layers perform the spherical correlations on SO(3) defined in Eq. 4. The rotation equivariance property specified in Eq. 5 ensures the spherical correlation result to have the same rotation property as the object.

To learn the unknown spherical representation f, the segmentation features ϕ consistent with the object view appearance are required. As shown in Fig. 3, the hemisphere segmentation feature ϕ is mainly learned from the ground-truth segmentation mask and the unknown spherical representation f is mainly learned from the target rotations. We formulate learning the auxiliary spherical representation as minimizing the following target:

$$f^* = \min_{f} \left\| R_g - \underset{R_p}{\operatorname{argmax}} ([\phi * f](R_p)) \right\|, \tag{6}$$

where R_p and R_g are the predicted and ground-truth rotations, respectively. Eq. 6 aims at learning the spherical representation f^* by minimizing the difference between R_p and R_g , where R_p is obtained by finding the rotation that maximizes the spherical correlation.

During training, the spherical representation is updated at each iteration mainly supervised by the ground-truth rotation R_g with a cross-entropy loss:

$$L_s = -log([\phi * f](R_g)).$$
(7)

Note that the spherical correlation result needs to be normalized first before computing the loss. Finally, we take a coarse rotation estimation by maximizing the spherical correlation result. In detail, we do max-pooling on the features from the spherical correlation, which is different from [12] that uses the SO(3)integration to obtain the full rotation invariance. The max-pooling on SO(3)space makes the inference sparse. The network goes to non-convergence when the rotation supervision is unavailable.

3.2. Feature Sampling

Observing the depth map containing noise, considering the scenario where the depth map is unavailable, the mapping from the image to the 3D graph is based on 2D sampling. Inside the object's segmentation mask, Poisson disc sampling [77] is applied and a fixed number of coordinates are collected.

During training, the segmentation branch is supervised by semantic labels. The feature maps before the last output are used for feature extraction. The coordinates collected through Poisson disk sampling [77] are rescaled into feature maps from different layers according to their times of pooling and unpooling. Since the collected coordinates are continuous, linear interpolation is applied along the feature channel. After the sampling of the 2D convolutional feature maps, the spherical features are sampled according to their 2D correspondences. The 2D convolutional features are combined with the spherical features which will be fed to the next stage.

3.3. Graph Building and Filling

Assume that a 3D mesh is a collection of vertices and edges that defines a 3D structure, it can be represented as a graph model G = (V, E). V is the set of nodes, and E is the set of edges. The graph convolutional layer can be written as a nonlinear function $H^{l+1} = f(H^l)$, where $H^0 = X$ is the input of the first layer, $X \in \mathbb{R}^{N \times D}$, N is the number of nodes in the graph, D is the dimension of the feature vector of each node. According to this definition, our method relies on [78] to handle 3D geometry. On an irregular graph, a graph convolutional layer is defined as:

$$H_u^{l+1} = \sigma(w_0 H_u^l + \sum_{v \in A(u)} w_1 H_v^l),$$
(8)

where u and v are the vertices of the graph model, $\sigma(\cdot)$ denotes a nonlinear activation function, A(u) indicates the neighboring vertices of u, w_0 and w_1 are learnable parameter matrices, and w_1 is shared for all edges.

In [9, 28], the 3D points are corresponding to depth map pixels which are aligned with the image. In our system, GCN is more flexible where graph pooling and unpooling can be applied. Because the graph is required to cover all objects in the image, the number of vertices of the graph may not be the same as the 2D sampling features. We thus consider two cases of constructing the graph with and without depth maps. Thereupon, a feature merging method based on graph matching is proposed.



Figure 4: Our graph can be initialized by either object 3D points or a plane with average depth if the depth map is unavailable.

For each instance point sampling and mapping, as shown in Fig. 4, we recover the 3D point of the object through each pixel and the known intrinsic matrix if the depth map is available. We define a square region with a set of uniformly distributed nodes as a regular grid. By connecting the nodes, the graph is initialized and is re-scaled as the inscribed polygon of the sampled 2D features. The static graph is used for the inference in which the graph structure is fixed but the features are filled to it. For the mapping from the 2D features to the graph, each vertex is first connected to the top N closest 2D points on the image. Then, consisting of a bipartite graph by the 3D points connected to the 2D points, we apply graph matching to make the features better distributed on the sphere. Finally, each 3D point retains only one assigned connection, and the corresponding 2D features are filled to the vertices.

If there is no depth map, it is impossible to directly recover the 3D points of the object. Similarly, we will construct the graph by sampling uniformly on the segmentation regions. Afterward, fill the vertices by finding the top N closest points of the vertices. To initialize 3D positions of the graph vertices, an average depth of the training dataset is used and the graph is initialized as a plane. Finally, the graph is also refined through graph matching.

For building the inference graph, a bipartite graph matching mechanism is used. Considering that the graph network is much harder going to convergence, the static graph is built. Meanwhile, for better training the network, the 2D features are sampled by Poisson disk sampling that introduces randomness to the features. During building the graph, the extracted feature points are assigned to their top N closest 3D mesh. To reduce the inconsistency between the static graph and randomly sampling, the bipartite graph matching mechanism is utilized for the feature merging. After the bipartite graph matching, the repeat assignments are cleared and the unique mapping is preserved.

Finally, according to the assignment, the 2D features from the convolutional network are extracted and filled into the graph vertices. Depending on the availability of the depth information, the vertex coordinates are initialized by either the plane with average depth or the points with depth. The final feature in each vertex is the concatenation of the convolutional feature and the vertex coordinate. After connecting the edges by the triangle mesh, the graph is built.

3.4. Graph Unpooling

A 3D mesh with rich details requires a large number of points to represent. As its abstracted representation, the graph complexity grows exponentially with the increasing number of mesh points. For better inference and reducing the data redundancy, graph unpooling is introduced into our network.



Figure 5: New vertices are first inserted in the middle of every edge. And then the new vertices (nodes) are connected by the Delaunay tesselation method.

The graph is initialized with fewer vertices. After several unpooling layers, a

sufficient number of vertices can be obtained. As indicated in Fig. 5, new vertices are inserted in the middle of every edge. Then new features are interpolated by two headers on each edge. Different from [70] where new connections are inside each triangle, after the interpolation, we rebuild the mesh by the Delaunay tesselation method for all the vertices. During the network inference, multiple stages of unpooling will be performed. After each graph unpooling, features from the convolutional layers will be mapped and concatenated to the new graph, which exploits the information from low-level layers. Finally, the highlevel graphs mix the feature from 2D convolution maps and the GCN inference.

In this way, the graph scale can be well controlled, while the graph network will easily go to convergence. Moreover, in the proposed system, the 3D shape is utilized for supervision. The dense output points are able to better fit the shape label, which helps the optimization of the network. Following the interpolation rule of unpooling, the number of points grows quickly. In our system, two graph unpooling layers are enough for obtaining a sufficient number of the final output points that provide further keypoint voting.

3.5. Losses

To better capture the spatial information of the 3D objects, in addition to predicting keypoints, the graph deforms the surface points of the object at the same time. During training, the graph is mainly supervised by the ground-truth object model points and the keypoints.

Graphic deformation helps to find the surface of the object more precisely. There is no point in calculating the absolute distance between two point sets. To measure the similarity of two distributions, Chamfer distance is more appropriate than the absolute distance [79, 80, 81]. Letting A and B denote as the two point sets, the Chamfer distance is defined as follows:

$$d_{CD}(A,B) = \frac{1}{|A|} \sum_{a \in A} \min_{b \in B} d(a,b) + \frac{1}{|B|} \sum_{b \in B} \min_{a \in A} d(a,b),$$
(9)

where d(a, b) denotes the distance between the two points (a, b), which is L_2 distance in our work, and |A| and |B| denote the numbers of points in sets A

and B, respectively. Thus, the Chamfer loss is computed using the distance.

$$L_{CD} = d_{CD}(P_{init}, P_{init} + t_{offset})$$

$$\tag{10}$$

where P_{init} are the points initialized in the graph vertices, and t_{offset} is the predicted deformation offset.

For keypoint learning, each vertex on the graph predicts the offset from the vertex position to each keypoint. We mainly refer to the L1 loss in [28] and configure it as follows:

$$L_{K} = \frac{1}{|P_{c}|} \sum_{p \in P_{c}} \sum_{k \in K} \left\| off(p,k) - off_{gt}(p,k) \right\|_{L^{1}},$$
(11)

where P_c is a set containing all the points belonging to category c. K is the keypoint set. off() and $off_{gt}(p,k)$ denotes the function computing the offset between the two points.

4. Experiment

4.1. Implementation Details

Our system extracts features from the segmentation branch. The segmentation network is of the encoder-decoder structure. It outputs a semantic segmentation map with N + 1 channels for N object categories and 1 negative channel. We directly use the trained segmentation network in [9, 82] but fine-tune the convolutional feature for the graph inference.

The spherical learning branch is built by the filters in the spherical correlation that contains one layer of S^2 correlation and four layers of SO(3) correlation. For the spherical learning, we use the angle grid size of 20 by default, although later we also test different grid sizes of the spherical correlation.

The graph convolutional network (GCN) is implemented fitting from [83]. The structure and parameter are shown in Table 1. The input graph has the feature dimension of 515 filled in 400 vertices. Expect for the first and the last one, the GCN layers are all of 192 dimension output. The last layer output provides the 3 dimensions of coordinates. First, a regular grid of $20 \times 20 = 400$

Layer Name	Output Dim.	Graph Nodes
GCN 1	512+3	400
GCN 2	192	400
GCN 3	192	400
Graph unpooling 1	192	3652
GCN 4	192	3652
GCN 5	192	3652
Graph unpooling 2	192	16147
GCN 6	3	16147

Table	1:	GCN	structure
-------	----	-----	-----------

points is initialized. Then we build the undirected graph using the 3D points as vertices and their mesh connections as edges, where the mesh is formed by the Delaunay tessellation method. As mentioned by [83], the graph is represented by its adjacency matrix. Chebyshev polynomial coefficients are fitted based on the Laplace matrix. In the inference process, GCN mainly predicts Chebyshev polynomial coefficients, and the final output is obtained through an inverse transformation. After 2 stages of graph unpooling, the final output graph scale contains 16147 nodes.

Our GCN branch also predicts the 3D object shape though the graph is initialized by the depth map or the plane. The supervision is built by the transformed 3D object models. Only the surface facing the camera is extracted, and a fixed number of points are sampled for the Chamfer distance loss function.

Depending on if the depth information is available or not, the graph can be initialized by either the points from the depth map or the plane with average depth. The average depth is obtained by calculating the mean object depth in the dataset. Results from two different initialization are compared.

For keypoint selection, we follow the settings in [25, 28]. Points on each object model are selected instead of predefined geometry points outside the model for more precise estimation. The output of the GCN represents the keypoints by the offsets from the transformed keypoints to the current model points. Then, the parameters are fitted according to the keypoints in a Hough Voting manner. The points are first transformed to their parameter space and inliers are used for the least-square fitting.

4.2. Evaluation Metrics

Most of the metrics for object 6D pose estimation can be divided into two categories. One focuses on the absolute precision of the pose parameters and the other focuses on measuring the intersection between the transformed or rendered object 3D models under the ground-truth and the predicted pose parameters, where the latter can consider the symmetry property of objects. [37, 7] introduce two metrics: Average Distance of Distinguishable (ADD) and its symmetry version, Average Distance of Distinguishable Symmetry (ADD-S). ADD computes the absolute distance between two transformed 3D model point sets:

$$ADD = \frac{1}{|V|} \sum_{v \in V} |(R_g v + t_g) - (R_p v + t_p)|,$$
(12)

where V denotes the 3D model point set and |V| is the number of points in the set. $R_{g,p}$ and $t_{g,p}$ are the ground-truth and predicted rotation and translation. And ADD-S considers the symmetry case by replacing the absolute distance by an average minimum distance which is defined as:

$$ADD - S = \frac{1}{|V|} \sum_{v_1 \in V} \min_{v_2 \in V} |(R_g v_1 + t_g) - (R_p v_2 + t_p)|.$$
(13)

Following [7, 9, 28], we calculate the Area Under Curve (AUC), the area under the threshold curve, for the two average distance, ADD and ADD-S. For YCB-Video, both the ADD and ADD-S AUC are compared. And only the ADD is computed for LINEMOD.

	PoseCNN	DF	PVN3D	Ours
	(ICP)	(iterative)	(ICP)	(depth)
002_master_chef_can	95.8	96.4	95.2	93.2
003_cracker_box	92.7	95.8	94.4	95.6
004_sugar_box	98.2	97.6	97.9	98.3
005_tomato_soup_can	94.5	94.5	95.9	97.5
006_mustard_bottle	98.6	97.3	98.3	97.9
007_tuna_fish_can	97.1	97.1	96.7	96.5
008_pudding_box	97.9	96.0	98.2	98.3
009_gelatin_box	98.8	98.0	98.8	97.8
010_potted_meat_can	92.7	90.7	93.8	93.3
011_banana	97.1	96.2	98.2	97.9

019_pitcher_base	97.8	97.5	97.6	97.9
021_bleach_cleanser	96.9	95.9	97.2	96.7
024_bowl*	81.0	89.5	92.8	94.3
025_mug	94.9	96.7	97.7	97.6
035_power_drill	98.2	96.0	97.1	95.9
036_wood_block*	87.6	92.8	91.1	94.1
037_scissors	91.7	92.0	95.0	95.5
040_large_marker	97.2	97.6	98.1	98.1
051_large_clamp*	75.2	72.5	95.6	95.6
052_extra_large_clamp*	64.4	69.9	90.5	91.1
061_foam_brick*	97.2	92.0	98.2	98.3
Average	93.0	93.2	96.1	96.3
	PoseCNN	DF	PVN3D	Ours
	(ICP)	(iterative)	(ICP)	(depth)
002_master_chef_can	68.1	73.2	79.3	70.9
003_cracker_box	83.4	94.1	91.5	93.5
004_sugar_box	97.1	96.5	96.9	97.9
005_tomato_soup_can	81.8	85.5	89.0	90.1
006_mustard_bottle	98.0	94.7	97.9	91.1
007_tuna_fish_can	83.9	81.9	90.7	90.2
008_pudding_box	96.6	93.3	97.1	97.9
009_gelatin_box	98.1	96.7	98.3	98.3
010_potted_meat_can	83.5	83.6	87.9	85.1
011_banana	91.9	83.3	96.0	96.9
019_pitcher_base	96.9	96.9	96.9	97.5
021_bleach_cleanser	92.5	89.9	95.9	93.9
024_bowl*	81.0	89.5	92.8	94.3
025_mug	81.1	88.9	96.0	95.3
035_power_drill	97.7	92.7	95.7	93.7
036_wood_block*	87.6	92.8	91.1	94.1
037_scissors	78.4	77.9	87.2	90.2
040_large_marker	85.3	93.0	91.6	93.6
051_large_clamp*	75.2	72.5	95.6	95.6
$052_extra_large_clamp*$	64.4	69.9	90.5	89.1
061_foam_brick*	97.2	92.0	98.2	98.3
Average	85.4	86.1	92.3	92.7

Table 2: The quantitative evaluations on YCB-Video. Methods (DF(iterative) [9], PoseCNN+ICP [7], PVN3D+ICP [28]) using depth information are compared. AUCs in ADD and ADD-S are computed. "*": symmetric objects.

4.3. Results on YCB-Video

The YCB-Video dataset is introduced in [7] which contains 21 objects with overall of 133,827 images. Following the dataset split in [7], the 80 videos with 80,000 synthetic images are for training and 2,949 keyframes are for testing. For convenience, both ADD and ADD-S AUC [7] are used for testing.

As shown in Table 2, AUC under ADD and ADD-S distance is evaluated for all the 21 objects in the dataset, YCB-Video. ADD-S is specifically designed for symmetric objects. Following settings in [7, 28], five symmetric objects are evaluated using ADD-S. For comparing with most of the best results on the YCB-Video dataset, we mainly refer to the RGBD methods or the RGB methods with ICP refinement which both consider the depth as the available information. The graph in our system is initialized by the points transformed from the depth map. We don't design any iterative refinement for our system. However, the proposed method reaches comparable performance with PVN3D+ICP [28]. And it outperforms the DenseFusion [9] method with iterative refinement. Since our method predicts the object surface at the same time while doing the keypoint estimation, more information is learned by our approach. We consider that the combination of the surface deformation and the keypoint prediction in multiple unpooling stages is like an increment refinement that is implicitly learned during the training stage. So we don't use an ICP for comparison here.

In the YCB-Video dataset, five objects are treated as symmetric objects, which are "024_bowl", "036_wood_block", "051_large_clamp", "052_extra_large_clamp" and "061_foam_brick". "025_mug" is an object that is of the non-symmetric 3D model but may appear symmetry when some of its parts are occluded. With better rotation prediction, the proposed method obtains consistent performance improvement on these objects.

Another challenge of the YCB-Video dataset is the confusion of the object detector for the classification between "051_large_clamp" and "052_extra_large_cl-amp" in the YCB-Video dataset. We give some visualization results in Fig. 6. For Fig. 6 (a), two clamps are classified as "051_large_clamp", in which one of them is wrongly classified. However, our result without ICP refinement shows



Figure 6: Results visualization on the YCB-Video dataset. Images from top to bottom in each column are an original image, the instance segmentation result, the object pose visualization and the object pose visualization with ICP refinement, respectively, where the pose visualizations are generated by projecting 3D models to their 2D shapes.

proper pose prediction for the wrongly predicted class, which suggests that our method has learned the proper visual feature from the image appearance.

4.4. Results on LINEMOD

We evaluate the proposed system on the LIEMOD [37] dataset with and without the depth information. The LINEMOD dataset contains 13 objects in 13 videos in extreme background clutter. We create the training set following [25] that crops object masks under ground-truth poses in each image and render them to different background images from PASCAL VOC [84]. And the training and testing split are following [7].

	RGB					
	DeepIM	PVNet	CDPN	Ours(plane)	-	
ape	77.0	43.6	64.4	75.6	-	
benchvise	97.5	99.9	97.8	93.2	-	
camera	93.5	86.9	91.7	93.5	-	
can	96.5	95.5	95.9	91.3	-	
cat	82.1	79.3	83.8	81.2	-	
driller	95.0	96.4	96.2	96.2	-	
duck	77.7	52.6	66.8	73.1	-	
eggbox	97.1	99.2	99.7	99.9	-	

99.4	95.7	99.6	100.0	-
52.8	82.0	85.8	83.3	-
98.3	98.9	97.9	97.1	-
97.5	99.3	97.9	98.1	-
87.7	92.4	90.8	91.1	-
88.6	86.3	89.9	90.3	-
		RGBD		
Point-Fusion	DF	DF	PVN3D	Ours
1 onte-1 usion	(perpixel)	(iterative)	TVINOE	(depth)
70.4	79.5	92.3	97.3	96.3
80.7	84.2	93.2	99.7	99.7
60.8	76.5	94.4	99.6	99.6
61.1	86.6	93.1	99.5	99.3
79.1	88.8	96.5	99.8	99.6
47.3	77.7	87.0	99.3	99.3
63.0	76.3	92.3	98.2	98.9
99.9	99.9	99.8	99.8	99.9
99.3	99.4	100.0	100.0	100.0
71.8	79.0	92.1	99.9	99.9
83.2	92.1	97.0	99.7	99.7
62.3	92.3	95.3	99.8	99.9
78.8	88.0	92.8	99.5	99.7
73.7	86.2	94.3	99.4	99.4
	99.4 52.8 98.3 97.5 87.7 88.6 Point-Fusion 70.4 80.7 60.8 61.1 79.1 47.3 63.0 99.9 99.3 71.8 83.2 62.3 78.8 73.7	$\begin{array}{c c} 99.4 & 95.7 \\ 52.8 & 82.0 \\ 98.3 & 98.9 \\ 97.5 & 99.3 \\ 87.7 & 92.4 \\ \hline \end{array} \\ \hline \end{array} \\ \hline \begin{array}{c} 88.6 & 86.3 \\ \hline \end{array} \\ \hline \end{array} \\ \hline \end{array} \\ \hline \begin{array}{c} Point-Fusion & DF \\ (perpixel) \\ \hline \end{array} \\ \hline \end{array} \\ \hline \begin{array}{c} 70.4 & 79.5 \\ 80.7 & 84.2 \\ \hline \end{array} \\ \hline \begin{array}{c} 60.8 & 76.5 \\ \hline 61.1 & 86.6 \\ \hline \end{array} \\ \hline \begin{array}{c} 79.1 & 88.8 \\ 47.3 & 77.7 \\ \hline \end{array} \\ \hline \begin{array}{c} 63.0 & 76.3 \\ 99.9 & 99.9 \\ 99.3 & 99.4 \\ \hline \end{array} \\ \hline \begin{array}{c} 71.8 & 79.0 \\ \hline \end{array} \\ \hline \begin{array}{c} 83.2 & 92.1 \\ \hline \end{array} \\ \hline \begin{array}{c} 62.3 & 92.3 \\ \hline \end{array} \\ \hline \begin{array}{c} 73.7 & 86.2 \\ \hline \end{array} \end{array}$	99.495.799.6 52.8 82.0 85.8 98.3 98.9 97.9 97.5 99.3 97.9 87.7 92.4 90.8 88.6 86.3 89.9 88.6 86.3 89.9 RGBDPoint-FusionDFDF $(iterative)$ (iterative) 70.4 79.5 92.3 80.7 84.2 93.2 60.8 76.5 94.4 61.1 86.6 93.1 79.1 88.8 96.5 47.3 77.7 87.0 63.0 76.3 92.3 99.9 99.9 99.8 99.9 99.9 99.8 99.3 99.4 100.0 71.8 79.0 92.1 83.2 92.1 97.0 62.3 92.3 95.3 78.8 88.0 92.8	99.4 95.7 99.6 100.0 52.8 82.0 85.8 83.3 98.3 98.9 97.9 97.1 97.5 99.3 97.9 98.1 87.7 92.4 90.8 91.1 88.6 86.3 89.9 90.3 88.6 86.3 89.9 90.3 RGBDPoint-FusionDFDF $promt-Fusion$ DF 92.3 97.3 97.3 80.7 84.2 93.2 99.7 60.8 76.5 94.4 99.6 61.1 86.6 93.1 99.5 79.1 88.8 96.5 99.8 47.3 77.7 87.0 99.3 63.0 76.3 92.3 98.2 99.9 99.9 99.8 99.8 99.3 99.4 100.0 100.0 71.8 79.0 92.1 99.9 83.2 92.3 95.3 99.8 78.8 88.0 92.8 99.5

Table 3: The quantitative evaluations on the LINEMOD dataset. Methods only based on RGB image (DeepIM [57], PVNet [25], CDPN [27]) and methods using the depth information (Point-Fusion [9], DF (perpixel) [9], DF (iterative) [9], PVN3D [28]) are compared.

As shown in Table 3, for the LINEMOD dataset, we evaluate the proposed method with and without the depth information. Initialized by a plane of average object depth in the dataset, the evaluation inference of the system does not need the depth map. Only the transformed object points are required for training supervision. Observing the performance decreasing when not using the depth map, we may consider that the system degenerates to the 2D coordinate map method. Without the depth information, the graph network predicts the translation parameter as well as the depth map. However, the graph is doing deformation in 3D space where the vertices relationship is captured. So, the proposed method still achieves comparable results with the state-of-the-art three methods [57, 25, 27]. Since there is a gap between synthetic training samples, the system performance can be further improved by filling the gap.

In Table 3, we reach the state-of-the-art result comparing with other methods using the depth map. Most of our basic modules are from DF [9]. After the graph matching and feature merging, the proposed method outperforms DF(per-pixel) and DF(iterative) [9] over 10 and 4 percentage points respectively where we don't apply any iterative process. We also reach comparable performance with PVN3D [28] which performs almost full correct rate under the ADD metric. However, we only do the segmentation from the 2D feature map while PVN3D [28] does it in 3D space.

4.5. Ablation Study

In this section, we conduct experiments to test different modules and settings in the proposed system, *i.e*let@tokeneonedotthe spherical learning part and the graph regression. We compare different variants of our method by removing and modifying the modules in different ways.

Before the ablation study on pose estimation, we first conduct an experiment on the rotated MNIST dataset to evaluate the effectiveness of spherical learning. Following [12], we rotate MNIST images on their spherical projection. Different combinations of non-rotated (NR) and rotated (R) sets for training and testing are considered. Similar to [12], we use a simple SphereCNN with two convolution layers. For learning the image rotation from the non-rotated training set, we do max-pooling on the features from the spherical correlation, which is different from [12] that uses the SO(3) integration to obtain the full rotation invariance. The pooled feature is considered to be rotation equivariance.

training/testing	NR/NR	R/R	NR/R
flat convolution	0.98	0.17	0.10
spherical learning	0.97	0.93	0.90

Table 4: Classification accuracy results on the synthetic MNIST dataset. NR/R refers to that the models are trained on the non-rotated set and tested on the rotated set.

Table 4 reports similar results as those in [12]. The pooled spherical features help obtain rotation-invariant results for different rotated inputs, which indicates the feasibility of our proposed approach to learn the rotation parameter from the spherical representation.

	MLP	G_{10}	G_{20}	G_{60}	G_{120}	PVN3D
024_bowl*	90.5	85.4	94.3	93.5	92.1	92.8
036_wood_block*	89.0	83.5	94.1	93.9	90.5	91.1
051_large_clamp*	92.7	89.1	95.6	95.9	94.6	95.6
$052_extra_large_clamp*$	86.5	90.3	91.1	90.5	89.1	90.5
061_foam_brick*	95.1	93.7	98.3	97.7	97.5	98.2

Table 5: Spherical learning ablation study.

First, we mainly test and compare the algorithm efficiency on symmetric objects in the YCB-Video that are "024_bowl", "036_wood_block", "051_large_cl-amp", "052_extra_large_clamp" and "061_foam_brick". Their AUCs under ADD-S are evaluated and the results are showing in Table 5.

In this experiment, we compare the spherical learning with direct regression by the Multi-Layer Perceptron (MLP). And the outputs with different grids are tested. The five symmetric objects in YCB-Video are used for testing. With all the other modules of the system remaining, only the spherical learning branch is changed. In Table 5, the ADD-S AUC results are shown. For all of the symmetric objects, using the spherical learning branch significantly improves the rotation regression.

We compare the results of our method under different output angle grid resolutions for the spherical learning branch. Particularly, we consider the grid sizes of 10, 20, 60 and 120 denote the corresponding variants of our method as G_{10} , G_{20} , G_{60} and G_{120} , respectively. It can be seen that 20 - 60 is a nosensitive range of the output grid. Inside this grid range, the performance of the spherical learning branch is not sensitive to its grid size. Thus, for all other experiments, we use G_{20} for efficiency. Meanwhile, the results from PVN3D [28] for the 5 symmetric objects are also compared which shows the efficiency of our proposed method on symmetric objects. Then, for better evaluating the performance of the proposed spherical learning module, we conduct the experiment combining the spherical learning standalone with the GCN branch. However, we use a keypoint-based regression instead of the GCN branch. The regression has the same output of the predefined keypoints.

	direct regression	$w/o sl^*$	w/ sl^*
YCB-Video	81.9	87.1	93.9
LINEMOD	70.2	77.6	85.1

Table 6: Ablation study for our method with/without the spherical learning combining the pose refinement modules. We report the ADD-S AUC on YCB-Video with depth input and LINEMOD without depth input over all the dataset objects. "sl*" represents the spherical learning module.

Table 6 shows the results of the ADD-S AUC on YCB-Video with depth input and LINEMOD without depth input over all the dataset objects. It can be seen that compared with the direct regression which directly regresses the pose parameter, both the spherical learning and the keypoint-based regression lead to better performance.

	Coarse	Refine
YCB-Video (ADD-S)	91.1	96.3
YCB-Video (ADD)	79.5	92.7

Table 7: Ablation study for coarse rotation learning.

In Table 7, we extract the rotation results from the coarse rotation estimation directly. From the results on YCB-Video, the performance deteriorates both under the ADD-S and ADD metrics. Due to the inaccurate of the coarse rotation estimation, the performance under ADD metric decreases a lot while the ADD-S result can maintain by satisfying translation estimation.

Then, we test different regression components of the proposed system. If both of the spherical learning and graph modules are removed, our method degrades to the baseline method, which directly regresses the pose parameters. For our method without the refinement module, the coarse rotation from the spherical learning is used as an incremental estimation that is added to the output of the baseline (direct parameter regression). For our method without the spherical learning part, we treat the output of the baseline as the coarse rotation estimation.

	CNN	CNN	graph	graph
	(parameter)	(keypoints)	(parameter)	(keypoints)
YCB-Video	85.9	93.6	88.3	96.3
LINEMOD	71.1	85.9	83.2	90.3

Table 8: Ablation study for the keypoint regression.

In Table 8, the comparison using different regression components is illustrated. Following the dataset settings of previous experiments, we test the ADD-S AUC on YCB-Video with depth input and LINEMOD without depth input. Besides the keypoint prediction, direct pose parameter regression is compared. For the direct pose regression, we predict the 6D pose parameter on the output feature maps. After an average pooling, the final pose is obtained. From Table 8, the keypoint prediction improves the overall accuracy both on CNN and the graph module. Meanwhile, with the spatial projection, the graph module can further improve the performance.

	Closest matching	bipartite graph matching
YCB-Video	93.0	96.6
LINEMOD	89.5	90.3

Table 9: Ablation study for bipartite graph matching.

The last experiment in Table 9 demonstrates the function of the bipartite graph matching. Without the bipartite graph matching, the nearest assignment from the convolution feature to the graph introduces repeat and missing points which makes the feature assignment become aliasing in the spatial space. With a bipartite graph matching to make an average of the assignment provides better feature alignment which obtains better results as shown in Table 9.

5. Conclusion

Considering the strength of Graph Convolutional Network (GCN) in finding the relationships among vertices, we abstract the 3D mesh as the graph and use a deep neural network to predict keypoints during the inference. For better convergence of the GCN, the static graph is built and the Poisson disc sampling is used to extract 2D points. The Poisson disc sampling introduces randomness into the network training so that the final output is more robust. Furthermore, to reduce the inconsistency of the static graph and the random point sampling, a bipartite graph matching mechanism is introduced into our system. Meanwhile, the 3D shape is used as supervision in the proposed training scheme. The graph unpooling is utilized for better fitting the 3D shape label, which also helps the low-level feature forward and the feature voting.

We have explored the feasibility of using spherical learning to alleviate the rotation symmetry problem in the object 6D pose estimation task. A spherical representation of an object is a natural representation inheriting the symmetry property of the object. Our proposed approach estimates the object rotation based on the spherical correlation in deep neural networks with spherical learning. On two very challenging datasets, we have compared our method with the state-of-the-art methods. Our system achieves stable performance improvement for most of the objects, which demonstrates the feasibility and effectiveness of the proposed approach.

References

- J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al., Recent advances in convolutional neural networks, Pattern Recognition 77 (2018) 354–377.
- [2] W. Liu, Z. Wu, H. Ding, F. Liu, J. Lin, G. Lin, Few-shot segmentation with global and local contrastive learning, arXiv preprint arXiv:2108.05293.
- [3] X. Sun, H. Ding, C. Zhang, G. Lin, K.-V. Ling, M2iosr: Maximal mutual information open set recognition, arXiv preprint arXiv:2108.02373.

- [4] J. Lin, Y. Cai, X. Hu, H. Wang, Y. Yan, X. Zou, H. Ding, Y. Zhang, R. Timofte, L. Van Gool, Flow-guided sparse transformer for video deblurring, arXiv preprint arXiv:2201.01893.
- [5] Y. Wang, Y. Cai, Y. Liang, H. Ding, C. Wang, S. Bhatia, B. Hooi, Adaptive data augmentation on temporal graphs, Advances in Neural Information Processing Systems 34.
- [6] Y. Hu, J. Hugonot, P. Fua, M. Salzmann, Segmentation-driven 6d object pose estimation, in: IEEE Conf. Comput. Vis. Pattern Recog., 2019, pp. 3385–3394.
- [7] Y. Xiang, T. Schmidt, V. Narayanan, D. Fox, Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, in: H. Kress-Gazit, S. S. Srinivasa, T. Howard, N. Atanasov (Eds.), Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June, 2018.
- [8] M. Rad, V. Lepetit, BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth, in: Int. Conf. Comput. Vis., 2017, pp. 3848–3856.
- [9] C. Wang, D. Xu, Y. Zhu, R. M. Martin, C. Lu, L. Fei-Fei, S. Savarese, Densefusion: 6d object pose estimation by iterative dense fusion, in: IEEE Conf. Comput. Vis. Pattern Recog., 2019, pp. 3343–3352.
- [10] S. Wang, K.-H. Yap, H. Ding, J. Wu, J. Yuan, Y.-P. Tan, Discovering human interactions with large-vocabulary objects via query and multi-scale detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 13475–13484.
- [11] M. Sundermeyer, Z. Marton, M. Durner, M. Brucker, R. Triebel, Implicit 3d orientation learning for 6d object detection from RGB images, in: Eur. Conf. Comput. Vis., 2018, pp. 712–729.
- [12] T. S. Cohen, M. Geiger, J. Köhler, M. Welling, Spherical cnns, in: Int. Conf. Learn. Represent., OpenReview.net, 2018.
- [13] C. Esteves, C. Allen-Blanchette, A. Makadia, K. Daniilidis, Learning SO(3) equivariant representations with spherical cnns, in: Eur. Conf. Comput. Vis., 2018, pp. 54–70.

- [14] D. G. Lowe, Object recognition from local scale-invariant features, in: Int. Conf. Comput. Vis., 1999, pp. 1150–1157.
- [15] H. Zhang, H. Ding, Prototypical matching and open set rejection for zero-shot semantic segmentation, in: Int. Conf. Comput. Vis., 2021, pp. 6974–6983.
- [16] C. Zhang, H. Ding, G. Lin, R. Li, C. Wang, C. Shen, Meta navigator: Search for a good adaptation policy for few-shot learning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9435–9444.
- [17] H. Ding, X. Jiang, A. Q. Liu, N. M. Thalmann, G. Wang, Boundary-aware feature propagation for scene segmentation, in: Int. Conf. Comput. Vis., 2019, pp. 6819– 6829.
- [18] H. Ding, X. Jiang, B. Shuai, A. Q. Liu, G. Wang, Semantic segmentation with context encoding and multi-path decoding, IEEE Trans. Image Process. 29 (2020) 3520–3533.
- [19] J. Liu, H. Ding, A. Shahroudy, L.-Y. Duan, X. Jiang, G. Wang, A. C. Kot, Feature boosting network for 3d pose estimation, IEEE transactions on pattern analysis and machine intelligence 42 (2) (2019) 494–501.
- [20] Y. Cai, Y. Wang, Y. Zhu, T.-J. Cham, J. Cai, J. Yuan, J. Liu, C. Zheng, S. Yan, H. Ding, et al., A unified 3d human motion synthesis model via conditional variational auto-encoder, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 11645–11655.
- [21] T. Li, Q. Ke, H. Rahmani, R. E. Ho, H. Ding, J. Liu, Else-net: Elastic semantic network for continual action recognition from skeleton data, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 13434– 13443.
- [22] A. Kundu, Y. Li, J. M. Rehg, 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare, in: IEEE Conf. Comput. Vis. Pattern Recog., 2018, pp. 3559–3568.
- [23] J. Mei, H. Ding, X. Jiang, Object 6d pose estimation with non-local attention, in: International Conference on Digital Image Processing, Vol. 11519, 2020, p. 115191H.

- [24] A. Saxena, J. Driemeyer, A. Y. Ng, Learning 3-d object orientation from images, in: IEEE Int. Conf. on Robotics and Automation, IEEE, 2009, pp. 794–800.
- [25] S. Peng, Y. Liu, Q. Huang, X. Zhou, H. Bao, Pvnet: Pixel-wise voting network for 6dof pose estimation, in: IEEE Conf. Comput. Vis. Pattern Recog., 2019, pp. 4561–4570.
- [26] A. Mousavian, D. Anguelov, J. Flynn, J. Kosecka, 3d bounding box estimation using deep learning and geometry, in: IEEE Conf. Comput. Vis. Pattern Recog., 2017, pp. 5632–5640.
- [27] Z. Li, G. Wang, X. Ji, CDPN: coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation, in: IEEE Conf. Comput. Vis. Pattern Recog., 2019, pp. 7677–7686.
- [28] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, J. Sun, PVN3D: A deep point-wise 3d keypoints voting network for 6dof pose estimation, in: IEEE Conf. Comput. Vis. Pattern Recog., 2020, pp. 11629–11638.
- [29] X. Li, H. Ding, Z. Tong, Y. Wu, Y. M. Chee, Primitive3d: 3d object dataset synthesis from randomly assembled primitives, in: IEEE Conf. Comput. Vis. Pattern Recog., 2022.
- [30] C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: IEEE Conf. Comput. Vis. Pattern Recog., 2017, pp. 77–85.
- [31] P. Wang, Y. Liu, Y. Guo, C. Sun, X. Tong, O-CNN: octree-based convolutional neural networks for 3d shape analysis, ACM Trans. Graph. 36 (4) (2017) 72:1– 72:11.
- [32] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, J. Yuan, 3d hand shape and pose estimation from a single RGB image, in: IEEE Conf. Comput. Vis. Pattern Recog., CVF/IEEE, 2019, pp. 10833–10842.
- [33] A. Collet, M. Martinez, S. S. Srinivasa, The MOPED framework: Object recognition and pose estimation for manipulation, Robotics Res. 30 (2011) 1284–1306.

- [34] F. Rothganger, S. Lazebnik, C. Schmid, J. Ponce, 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints, Int. J. Comput. Vis. 66 (3) (2006) 231–259.
- [35] J. Ren, X. Jiang, J. Yuan, Learning lbp structure by maximizing the conditional mutual information, Pattern Recognition 48 (2015) 3180–3190.
- [36] Z. Miao, X. Jiang, Interest point detection using rank order log filter, Pattern Recognition 46 (2013) 2890–2901.
- [37] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. R. Bradski, K. Konolige, N. Navab, Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes, in: Asian Conference on Computer Vision, 2012, pp. 548–562.
- [38] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, C. Rother, Learning 6d object pose estimation using 3d object coordinates, in: Eur. Conf. Comput. Vis., 2014, pp. 536–551.
- [39] J. Li, Z. Feng, Q. She, H. Ding, C. Wang, G. H. Lee, Mine: Towards continuous depth mpi with nerf for novel view synthesis, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 12578–12588.
- [40] L. Bo, X. Ren, D. Fox, Learning hierarchical sparse features for RGB-(D) object recognition, Robotics Res. 33 (4) (2014) 581–599.
- [41] M. Schwarz, H. Schulz, S. Behnke, RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features, in: IEEE Int. Conf. on Robotics and Automation, 2015, pp. 1329–1335.
- [42] W. Kehl, F. Milletari, F. Tombari, S. Ilic, N. Navab, Deep learning of local RGB-D patches for 3d object detection and 6d pose estimation, in: Eur. Conf. Comput. Vis., 2016, pp. 205–220.
- [43] H. Ding, X. Jiang, B. Shuai, A. Q. Liu, G. Wang, Context contrasted feature and gated multi-scale aggregation for scene segmentation, in: IEEE Conf. Comput. Vis. Pattern Recog., 2018, pp. 2393–2402.

- [44] H. Ding, C. Liu, S. Wang, X. Jiang, Vision-language transformer and query generation for referring segmentation, in: Int. Conf. Comput. Vis., 2021, pp. 16321– 16330.
- [45] H. Ding, X. Jiang, B. Shuai, A. Q. Liu, G. Wang, Semantic correlation promoted shape-variant context for segmentation, in: IEEE Conf. Comput. Vis. Pattern Recog., 2019, pp. 8885–8894.
- [46] B. Shuai, H. Ding, T. Liu, G. Wang, X. Jiang, Toward achieving robust lowlevel and high-level scene parsing, IEEE Transactions on Image Processing 28 (3) (2018) 1378–1390.
- [47] Z. Tong, Y. Liang, H. Ding, Y. Dai, X. Li, C. Wang, Directed graph contrastive learning, Advances in Neural Information Processing Systems 34.
- [48] X. Wang, X. Jiang, H. Ding, J. Liu, Bi-directional dermoscopic feature learning and multi-scale consistent decision fusion for skin lesion segmentation, IEEE transactions on image processing 29 (2019) 3039–3051.
- [49] S. Wang, Y. Duan, H. Ding, Y.-P. Tan, K.-H. Yap, J. Yuan, Learning transferable human-object interaction detector with natural language supervision, in: IEEE Conf. Comput. Vis. Pattern Recog., 2022.
- [50] J. Mei, Z. Wu, X. Chen, Y. Qiao, H. Ding, X. Jiang, Deepdeblur: text image recovery from blur to sharp, Multimedia tools and applications 78 (13) (2019) 18869–18885.
- [51] C. Liu, X. Jiang, H. Ding, Instance-specific feature propagation for referring segmentation, IEEE Transactions on Multimedia.
- [52] C. Liu, H. Ding, X. Jiang, Towards enhancing fine-grained details for image matting, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 385–393.
- [53] M.-J. Chiou, H. Ding, H. Yan, C. Wang, R. Zimmermann, J. Feng, Recovering the unbiased scene graphs from the biased ones, in: Proceedings of the 29th ACM International Conference on Multimedia, 2021, pp. 1581–1590.

- [54] H. Ding, S. Cohen, B. Price, X. Jiang, Phraseclick: toward achieving flexible interactive segmentation by phrase and click, in: European Conference on Computer Vision, Springer, 2020, pp. 417–435.
- [55] X. Wang, H. Ding, X. Jiang, Dermoscopic image segmentation through the enhanced high-level parsing and class weighted loss, in: 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 245–249.
- [56] P. Poirson, P. Ammirato, C. Fu, W. Liu, J. Kosecka, A. C. Berg, Fast single shot detection and pose estimation, in: International Conference on 3D Vision, 2016, pp. 676–684.
- [57] Y. Li, G. Wang, X. Ji, Y. Xiang, D. Fox, Deepim: Deep iterative matching for 6d pose estimation, in: Eur. Conf. Comput. Vis., 2018, pp. 695–711.
- [58] F. Manhardt, D. M. Arroyo, C. Rupprecht, B. Busam, T. Birdal, N. Navab, F. Tombari, Explaining the ambiguity of object detection and 6d pose from visual data, in: Int. Conf. Comput. Vis., 2019, pp. 6841–6850.
- [59] N. J. Mitra, L. J. Guibas, M. Pauly, Partial and approximate symmetry detection for 3d geometry, ACM Trans. Graph. 25 (3) (2006) 560–568.
- [60] D. M. Thomas, V. Natarajan, Detecting symmetry in scalar fields using augmented extremum graphs, IEEE Trans. Vis. Comput. Graph. 19 (12) (2013) 2663–2672.
- [61] P. Speciale, M. R. Oswald, A. Cohen, M. Pollefeys, A symmetry prior for convex variational 3d reconstruction, in: Eur. Conf. Comput. Vis., 2016, pp. 313–328.
- [62] D. Raviv, A. M. Bronstein, M. M. Bronstein, R. Kimmel, Full and partial symmetries of non-rigid shapes, Int. J. Comput. Vis. 89 (1) (2010) 18–39.
- [63] D. Panozzo, Y. Lipman, E. Puppo, D. Zorin, Fields on symmetric surfaces, ACM Trans. Graph. 31 (4) (2012) 111:1–111:12.
- [64] V. G. Kim, Y. Lipman, T. A. Funkhouser, Blended intrinsic maps, ACM Trans. Graph. 30 (4) (2011) 79.

- [65] C. Li, M. Wand, X. Wu, H. Seidel, Approximate 3d partial symmetry detection using co-occurrence analysis, in: 2015 International Conference on 3D Vision, 3DV 2015, Lyon, France, October 19-22, 2015, 2015, pp. 425–433.
- [66] M. Lukác, D. Sýkora, K. Sunkavalli, E. Shechtman, O. Jamriska, N. Carr, T. Pajdla, Nautilus: recovering regional symmetry transformations for image editing, ACM Trans. Graph. 36 (4) (2017) 108:1–108:11.
- [67] R. Nagar, S. Raman, Fast and accurate intrinsic symmetry detection, in: Eur. Conf. Comput. Vis., 2018, pp. 433–450.
- [68] L. Zhao, X. Peng, Y. Tian, M. Kapadia, D. N. Metaxas, Semantic graph convolutional networks for 3d human pose regression, in: IEEE Conf. Comput. Vis. Pattern Recog., Computer Vision Foundation / IEEE, 2019, pp. 3425–3435.
- [69] S. Yan, Y. Xiong, D. Lin, Spatial temporal graph convolutional networks for skeleton-based action recognition, in: S. A. McIlraith, K. Q. Weinberger (Eds.), AAAI, AAAI Press, 2018, pp. 7444–7452.
- [70] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, Y.-G. Jiang, Pixel2mesh: Generating 3d mesh models from single rgb images, in: Eur. Conf. Comput. Vis., 2018, pp. 52–67.
- [71] C. Wen, Y. Zhang, Z. Li, Y. Fu, Pixel2mesh++: Multi-view 3d mesh generation via deformation, in: Int. Conf. Comput. Vis., IEEE, 2019, pp. 1042–1051.
- [72] B. Doosti, S. Naha, M. Mirbagheri, D. J. Crandall, Hope-net: A graph-based model for hand-object pose estimation, in: IEEE Conf. Comput. Vis. Pattern Recog., IEEE, 2020, pp. 6607–6616.
- [73] H. Ding, H. Zhang, J. Liu, J. Li, Z. Feng, X. Jiang, Interaction via bi-directional graph of semantic region affinity for scene parsing, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 15848– 15858.
- [74] H. Ding, H. Zhang, C. Liu, X. Jiang, Deep interactive image matting with feature propagation, IEEE Transactions on Image Processing 31 (2022) 2421–2432.

- [75] J. R. Driscoll, D. M. Healy, Computing fourier transforms and convolutions on the 2-sphere, Advances in applied mathematics.
- [76] C. G. S., A. B. Kyatkin, Engineering applications of noncommutative harmonic analysis: with emphasis on rotation and motion groups, CRC press, 2000.
- [77] Bridson, Robert, Fast poisson disk sampling in arbitrary dimensions, SIGGRAPH sketches 10 (2007) 1.
- [78] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: going beyond euclidean data, IEEE Signal Processing Magazine 34 (2017) 18–42.
- [79] J. Ribera, D. Guera, Y. Chen, E. J. Delp, Locating objects without bounding boxes, in: IEEE Conf. Comput. Vis. Pattern Recog., 2019, pp. 6479–6489.
- [80] O. Schütze, X. Esquivel, A. Lara, C. A. C. Coello, Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization, IEEE Trans. Evolutionary Computation 16 (2012) 504–522.
- [81] A. A. Taha, A. Hanbury, Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool, BMC Medical Imaging 15 (2015) 29.
- [82] X. Wang, X. Jiang, H. Ding, Y. Zhao, J. Liu, Knowledge-aware deep framework for collaborative skin lesion segmentation and melanoma recognition, Pattern Recognition 120 (2021) 108075.
- [83] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Int. Conf. Learn. Represent., OpenReview.net, 2017.
- [84] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, Int. J. Comput. Vis. 88 (2010) 303– 338.